VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA STROJNÍHO INŽENÝRSTVÍ
FACULTY OF MECHANICAL ENGINEERING

ÚSTAV VÝROBNÍCH STROJŮ, SYSTÉMŮ A ROBOTIKY
INSTITUTE OF PRODUCTION MACHINES, SYSTEMS AND ROBOTICS

VYUŽITÍ UMĚLÉ INTELIGENCE V TECHNICKÉ DIAGNOSTICE
UTILIZATION OF ARTIFICIAL INTELLIGENCE IN TECHNICAL DIAGNOSTICS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — **Bc. Antonín Konečný**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. Daniel Zuth, Ph.D.**
SUPERVISOR

BRNO 2021

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
FAKULTA STROJNÍHO INŽENÝRSTVÍ

Zadání diplomové práce

Ústav:	Ústav výrobních strojů, systémů a robotiky
Student:	**Bc. Antonín Konečný**
Studijní program:	Strojní inženýrství
Studijní obor:	Kvalita, spolehlivost a bezpečnost
Vedoucí práce:	**Ing. Daniel Zuth, Ph.D.**
Akademický rok:	2020/21

Ředitel ústavu Vám v souladu se zákonem č.111/1998 o vysokých školách a se Studijním a zkušebním řádem VUT v Brně určuje následující téma diplomové práce:

Využití umělé inteligence v technické diagnostice

Stručná charakteristika problematiky úkolu:

Práce se bude zabývat možností využití metod umělé inteligence pro vyhodnocení poruch strojního zařízení. Důležitou částí práce bude přehled a popis použitelných metod a následně aplikace některých metod na vzorku naměřených dat. Aplikace bude řešena v prostředí Python.

Cíle diplomové práce:

Rešerše v oblasti metod uměle inteligence.
Vybrat a popsat vhodné programovací prostředí (knihovny).
Vybrat a popsat vhodné metody klasifikace dat (nejméně 3).
Popsat zdrojová data pro následně zpracování.
Vytvořit aplikaci ve vybraném SW prostředí.
Vyhodnotit úspěšnost klasifikace včetně vizualizace.
Dokumentace vytvořeného řešení.

Seznam doporučené literatury:

DANIŠ, Stanislav. Základy programování v prostředí Octave a Matlab. Praha: Matfyzpress, 2009. ISBN 978-80-7378-082-1.

VONDRÁK, Ivo. Umělá inteligence a neuronové sítě. 3. vyd. Ostrava: VŠB - Technická univerzita Ostrava, 2009. ISBN 978-80-248-1981-5.

Termín odevzdání diplomové práce je stanoven časovým plánem akademického roku 2020/21

V Brně, dne

L. S.

doc. Ing. Petr Blecha, Ph.D. ředitel ústavu	doc. Ing. Jaroslav Katolický, Ph.D. děkan fakulty

ABSTRAKT

Diplomová práce je zaměřena na využití metod umělé inteligence pro vyhodnocení poruchového stavu strojního zařízení. Vyhodnocovaná data jsou z vibrodiagnostického modelu pro simulaci statické a dynamické nevývahy. Aplikovány jsou metody strojového učení, konkrétně učení s učitelem. V práci je popsáno softwarové prostředí Spyder, jeho alternativy, a programovací jazyk Python, ve kterém jsou skripty napsány. Obsahuje přehled s popisem použitých knihoven (`Scikit-learn`, `SciPy`, `Pandas` ...) a metod, kterými jsou klasifikace metodou nejbližšího souseda (KNN), metoda podpůrných vektorů (SVM), rozhodovací stromy (DT) a klasifikace metodou náhodného lesa (RF).

Výsledky úspěšnosti klasifikace jsou vizualizovány ve výsledné klasifikační matici pro každou metodu. Součástí přílohy jsou napsané skripty pro zpracování a výpočet prediktorů, hledání nejvhodnějších parametrů modelu, hodnocení úspěšnosti učení a klasifikace s vizualizací výsledku.

KLÍČOVÁ SLOVA

umělá inteligence, strojové učení, učení s učitelem, klasifikace, extrakce charakteristických rysů, Spyder, Python, Scikit-learn, sklearn, hledání nejvhodnějších parametrů, metoda nejbližšího souseda, metoda podpůrných vektorů, rozhodovací stromy, klasifikace metodou nejbližšího souseda, klasifikační matice

ABSTRACT

The diploma thesis is focused on the use of artificial intelligence methods for evaluating the fault condition of machinery. The evaluated data are from a vibrodiagnostic model for simulation of static and dynamic unbalances. The machine learning methods are applied, specifically supervised learning. The thesis describes the Spyder software environment, its alternatives, and the Python programming language, in which the scripts are written. It contains an overview with a description of the libraries (`Scikit-learn`, `SciPy`, `Pandas` ...) and methods – K-Nearest Neighbors (KNN), Support Vector Machines (SVM), Decision Trees (DT) and Random Forests Classifiers (RF).

The results of the classification are visualized in the confusion matrix for each method. The appendix includes written scripts for feature engineering, hyperparameter tuning, evaluation of learning success and classification with visualization of the result.

KEYWORDS

Artificial Intelligence, Machine Learning, Supervised Learning, Classification, Feature Engineering, Spyder, Python, Scikit-learn, sklearn, Hyperparameter Tuning, K-Nearest Neighbors, Support Vector Machines, Decision Trees, Random Forests Classifiers, Confusion Matrix

KONEČNÝ, Antonín. *Využití umělé inteligence v technické diagnostice*. Brno: Vysoké učení technické v Brně, Fakulta strojního inženýrství, Ústav výrobních strojů, systémů a robotiky, 2021, 77 s. Diplomová práce. Vedoucí práce: Ing. Daniel Zuth, Ph.D.

Prohlášení autora o původnosti díla

Jméno a příjmení autora:	Bc. Antonín Konečný
VUT ID autora:	191435
Typ práce:	Diplomová práce
Akademický rok:	2020/21
Téma závěrečné práce:	Využití umělé inteligence v technické diagnostice

Prohlašuji, že svou závěrečnou práci jsem vypracoval samostatně pod vedením vedoucí/ho závěrečné práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené závěrečné práce dále prohlašuji, že v souvislosti s vytvořením této závěrečné práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno ....

podpis autora*

*Autor podepisuje pouze v tištěné verzi.

PODĚKOVÁNÍ

Poděkovat by se, dle mého soudu, mělo každému! Jenže komu tedy děkovat. Vše, co v této práci uvádím, jsem se naučil nebo jsem poznal při studiu na VUT v Brně. Dík tak bezesporu patří všem, kdo se mě na universitě snažili vzdělat. Také knihovnám, které byly po celou dobu studentům k dispozici, jmenovitě areálové knihovně FSI a knihovně FIT, ze kterých jsem vypůjčoval literaturu k tvorbě této práce.

Těžiště díku samozřejmě náleží mému vedoucímu doktoru Danielu Zuthovi. Za jeho obětavé vedení v době „home officeové“, chcete-li „koronavirové“ jsme měli naštěstí ustálené spojení a kdykoliv jsem se zasekl nebo vznikl nějaký problém, během okamžiku mi přišla rada nebo nasměrování. Když jsme se před pěti lety poprvé setkali v předmětu Informatika, netušil jsem, že při dokončování magisterského studia budu rozvíjet své znalosti v oblasti umělé inteligence a technické diagnostiky pod jeho vedením. Máte mé díky a uznání.

Své díky dlužím prarodičům Marii a Antonínu Konečným, kteří mi po dobo studia na vysoké škole vytvořili dokonalé prostředí ve svém domě k nerušenému rozjímání o (nejen) technických problémech dnešní doby. Dodávali odhodlání a společně jsme se mohli těšit ze studijních úspěchů. O něco vzdálenější, ale o to větší hmotnou podporu, psychickou a morální jsem pociťoval od svých rodičů Drahomíry a Antonína. Dostatečně nedovedu poděkovat svému strýci Františku Hilbertovi, s kterým jsem strávil čas na korekci technické stránky práce a jeho ženě Marii za kontrolu pravopisu.

Dokonalým přátelům Ivě, Karlovi, Johance a Matějovi za probdělé noci, podporu a víru v mé schopnosti. Závěrem bych si vypůjčil jeden citát od Tomáše Sedláčka, který vystihuje konec mých slov: „Největší dík náleží tomu, jehož skutečné jméno neznám.“

Obsah

1 ÚVOD

Umělá inteligence (UI) je (sou)částí našich životů víc, než si někteří uvědomují. Tento fenomén velmi čtivě a jasně popisují například literatury [8, 9]. Neexistuje lidská činnost, u které bychom nedokázali s větší nebo menší výhodou aplikovat nějakou metodu UI. UI nám umožnila vyvinout dopravní prostředky bez řidiče, pomáhá zdokonalit překladače jazyků nebo plynulost konverzace s virtuálními asistenty (Siri, Alexa ...). Dovede hrát šach lépe než nejlepší hráči světa. Pomáhá odhalit pokusy o podvodné jednání nebo určit emoce ze způsobu vyjadřování. Navádění autonomních, robotických asistentů v domácnosti je také zásluhou UI nebo rozpoznání poruchového stavu a navíc i určení typu poruchy například v technické diagnostice. To je úkolem této diplomové práce.

Teoretická část práce je zaměřena na stručný popis technické diagnostiky. Jak se vyvíjela v průběhu času a že bez požadavku na kvalitu by nás spolehlivost, respektive diagnostikovatelnost, vůbec nezajímala. Je vysvětlen sled kroků, které vedou od diagnostikovaného objektu až k určení „diagnózy“. Těžiště celé práce je v umělé inteligenci, jíž se v rešerši autor věnuje nejvíce. Důležitou částí je vlastní definice UI a součásti – strojového učení, které je klíčové pro řešení problému klasifikace dat nejen v diagnostice. Je jasně popsána a definována metoda učení s učitelem, na kterou je zaměřena praktická část. Součástí rešerše jsou i softwarové alternativy, ve kterých by bylo možné práci řešit. Dále je popsáno softwarové prostředí a programovací jazyk Python, ve kterém je praktická část práce řešena, a popis základních knihoven, které jsou pro řešení použity.

Pro praktickou část je popsán diagnostický model (vibrodiagnostika), který je použit pro simulaci poruchových stavů (statická a dynamická nevývaha). Dále je v praktické části definován prediktor, jak se k němu dostat a které jsou použity. Je vysvětlen důvod, proč je výběr a počet prediktorů tak zásadní pro úspěšnost klasifikace.

Jednotlivé aplikované metody klasifikace dat (klasifikace metodou nejbližšího souseda, metoda podpůrných vektorů, rozhodovací stromy a klasifikace metodou náhodného lesa) jsou podrobně popsány i s výhodami a nevýhodami, které z jejich aplikace mohou nastat. Součástí hodnocení výsledků je grafická vizualizace pomocí výsledných klasifikačních matic pro jednotlivé metody.

Umělá inteligence nám dokáže nedocenitelně pomoci v řešení širokého portfolia problémů. Znalost metodiky tvorby UI od zpracování dat přes učení modelu a jeho následné ověření, se může stát pro budoucí inženýry nepostradatelnou základní vědomostí kterou, jak již bylo zmíněno, lze aplikovat kdekoliv!

2 TECHNICKÁ DIAGNOSTIKA

Obecně lze napsat, že v současné době s rostoucí složitostí veškerých systémů[1], k jejichž opravě nám nestačí kladivo, pár šroubováků a sada klíčů, roste i význam spolehlivého určení příčiny a místa poruchy. Požadavek na spolehlivost je neoddělitelnou součástí kvality produktu, jejíž vztah k diagnostikovatelnosti (technické diagnostice) bude popsán v kap. 2.1. Dále se [konečně – pozn. aut.] klade čím dál větší důraz na prevenci (předcházení) provozních problémů, které jsou způsobeny technickými závadami. Vývoj diagnostikovatelnosti směřoval od prostého hledání příčiny poruch, přes pravidelné kontroly, prohlídky a revize, ke stálému automatickému monitorování technického stavu strojů a strojních zařízení. Proto se z diagnostikovatelnosti stal samostatný vědní obor – technická diagnostika[2].[1]

2.1 Od kvality produktu k technické diagnostice

Dle ČSN EN ISO 9000:2016 je **kvalitou** myšleno „*Stupeň splnění požadavků souborem inherentních charakteristik objektu*“, kde *požadavkem* myslíme potřebu zákazníka nebo jeho očekávání a tento požadavek je obecně předpokládán nebo je závazný, *charakteristika* je jakýmsi znakem, rozlišující vlastností. *Inherentní* znamená vlastní, existující v něčem, zejména jako trvalá charakteristika.[2]

Z teorie výrobních procesů můžeme však **kvalitu** chápat jako „*Stupeň splnění požadavků vystihující podstatu výrobku*“, jinými slovy že výrobek mám tak kvalitní, jak kvalitní je jeho proces výroby. Jak mám tento proces *poznaný* [sledujme návaznost na překlad technické diagnostiky z řečtiny – pozn. aut.] a jak moc je opakovatelný, tak kvalitně budu vyrábět produkty. Tudíž skrze ono poznání – diagnostikování jsme schopni mnohem lépe zajistit kvalitu výrobku, procesu nebo technického zařízení.[3]

Díky vyznačené cestě (červeně) na obr. 2.1 a obr. 2.2 jsme schopni si lépe představit cestu od spolehlivosti až k diagnostikovatelnosti. Diagnostikováním poruchového stavu pak myslíme úkony, prováděné za účelem monitorování a identifikaci poruchového stavu, lokalizaci porouchané entity a identifikaci vady nebo poruchy. Diagnózou pak v technické praxi myslíme okamžité vyhodnocení technického stavu objektu, jinými slovy jde o vyhodnocení provozuschopnosti objektu za daných technických podmínek. Jak již bylo zmíněno,

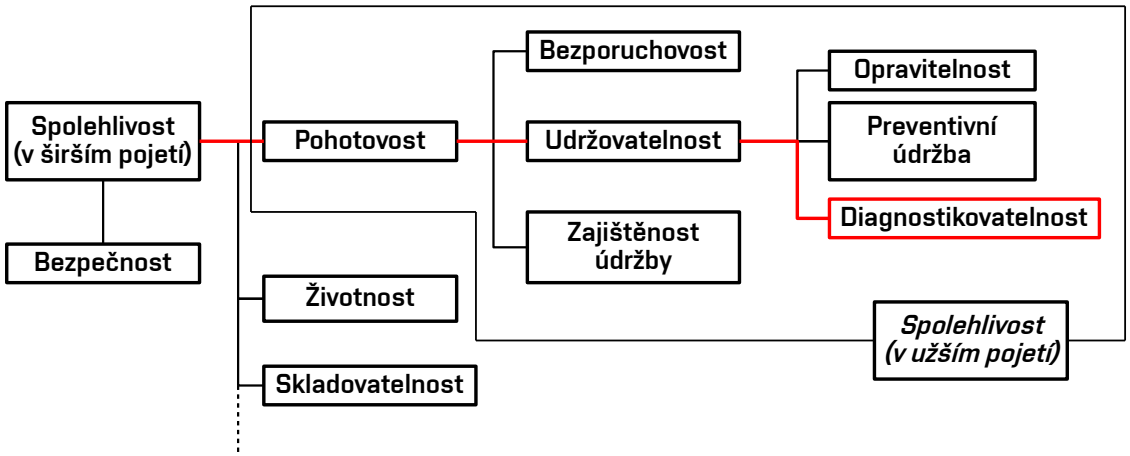
[1] Jako příklad uveďme autorův současný motocykl HODNA CB 500 roč.výr. 2001, kde palivová soustava neobsahuje žádnou elektroniku a při vzniku (diagnostikování) problému s přidáním plynu, bylo relativně jednoduché demontování celé soustavy (především karburátoru), její kontrola, čištění a opětovná instalace za použití běžného nářadí – problém vyřešen v pohodlí domova. Dnešní moderní motocykly mají mnoho elektronického vybavení a domácí demontáž některé části je spíše úspěšným pokusem o uvedení motocyklu do neprovozuschopného stavu. Bez potřebného vybavení, leckdy i napojení na diagnostiku systému, nejsme schopni bez odborné pomoci cokoliv sami udělat (opravit).

[2] Technická diagnostika vychází z řeckého slova DIA-GNOSIS – „*skrze poznání*“.[4]

Pojem *diagnóza* je převzat taktéž z řečtiny, kdy byl využíván ve smyslu lékařského vyšetření pacienta, mnohem později tento pojem dostává i technický rozměr.[5]

stala se technická diagnostika samostatným vědním oborem s využitím především bezdemontážních nedestruktivních metod k určení technického stavu objektu. Základními úkoly technické diagnostiky pak jsou detekce (identifikace) a lokalizace (umístění) vady nebo poruchy.[4]

Obr. 2.1: Určení kvality produktu a vyjádření užitných vlastností [4]



Obr. 2.2: Definice spolehlivosti v širším a užším pojetí [4]

V současné době je těžiště technické diagnostiky posouváno směrem k průběžnému monitorování technického stavu objektu – tzn. předcházení vzniku poruch. Signály[3]obecně, vznikající při provozu libovolného objektu, jsou především využívány k jeho řízení a zajištění správné funkce. Tyto signály však mohou být nositelem zásadních informací o tech-

[3]Signál je fyzikální veličina, která je obecně nositelem informace nebo informací o určité události.[4]

nickém stavu objektu. Pokud se nám podaří najít vhodné sledované parametry a správně je vyhodnotit, můžeme díky těmto, na první pohled „funkčním“ signálům, zjistit mnohé o technickém stavu objektu. Je zřejmé, že takto shromažďovaná data mají obrovský objem a k jejich zpracování a hodnocení je zapotřebí využít moderních přístupů, mezi kterými může být např. využití umělé inteligence.

2.2 Diagnostický prostředek, systém a signál

Diagnostickým prostředkem se rozumí souhrn technických zařízení (např. senzorů) a pracovních postupů určených k analýze a vyhodnocení technického stavu diagnostikovaného objektu. Pracovními postupy jsou algoritmy (úkony) a programové vybavení pro generování a vyhodnocení testů, metody výběru diagnostických veličin, sestavení matematických modelů atd. Diagnostické prostředky mohou být dvojího typu a to vnitřní (zabudované do objektu) nebo vnější (samostatné, demontovatelné a přemístitelné).[4]

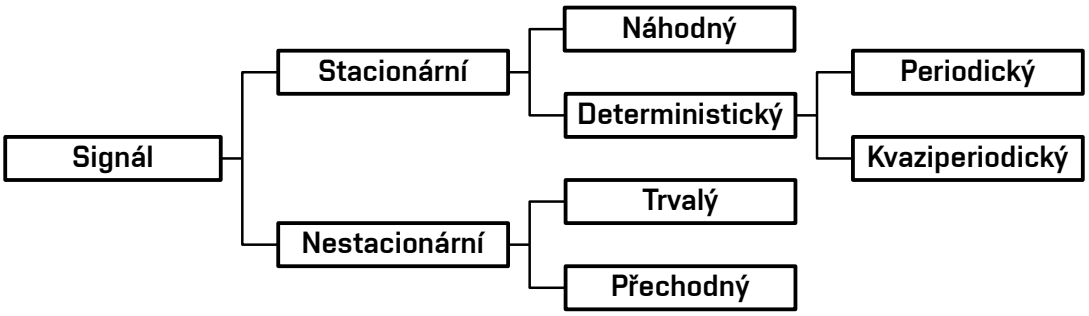
Diagnostický systém je pak tvořen diagnostickými prostředky, diagnostikovaným objektem a obsluhou. Diagnostické systémy můžeme dělit na: [4]

- **On-line** systémy vyhodnocují technický stav diagnostikovaného objektu při provozu. Rozšířenou podskupinou jsou tzv. *monitorovací systémy*, které jsou trvale připojeny k diagnostikovanému objektu a trvale sledují technický stav objektu a průběžně vyhodnocují mezní stavy objektu. Podstatnou část dnes tvoří tzv. *automatické diagnostické systémy*, které mají počítačovou podporu a jsou zpravidla napojeny na expertní diagnostické systémy.
- **Off-line** diagnostikování se provádí když je diagnostikovaný objekt mimo provoz. Dělí se dále na *nezávislé* – sled jednotlivých kroků je nezávislý na výsledcích předcházejících kroků, hodnocení je podmíněno provedením všech kroků testu, a *závislé* – realizuje kroky testu v závislosti na výsledcích předcházejících kroků.
- **Pokročilé diagnostické systémy** jsou kombinací *on-line* a *off-line* diagnostických systémů, v současné době nejvíce se rozšiřující systémy.

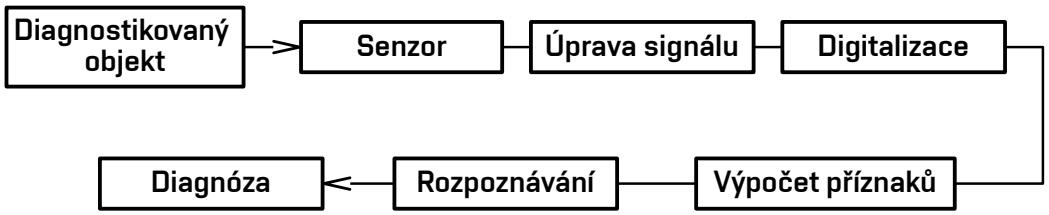
Diagnostický signál je fyzikální veličina, která nese specifickou informaci o diagnostické události. Rozdělení diagnostických signálů je na obr. 2.3. Pomocí vhodného zpracování diagnostického signálu jsme schopni určit diagnózu, tento postup se nazývá diagnostickým řetězcem obr. 2.4.[4]

Zaznamenávané diagnostické veličiny, pomocí senzoru, jsou převedeny na diagnostické signály. Následuje jejich úprava (přenos, zesílení, analogová filtrace aj.) a převod do číslicové podoby – digitalizace. Takto získaná data jsou často dále filtrována (vyhlazení, potlačení šumu, apod.). Následuje výpočet relevantních sledovaných parametrů (výpočet příznaků)[4]. Předposledním krokem v tomto řetězci je tzv. rozpoznání, jehož úkolem je rozpoznat stav objektu nebo procesu na základě naměřených dat. Tento krok často řeší

[4]V ojedinělých případech, kdy je výstupem předzpracovaný signál, který vyhodnocujeme např. pouze jednoduchým porovnáním amplitud s prahovou hodnotou, není blok výpočtů příznaků potřeba.[4]



Obr. 2.3: Rozdělení diagnostického signálu [4]



Obr. 2.4: Obecné uspořádání automatizovaného diagnostického řetězce [4]

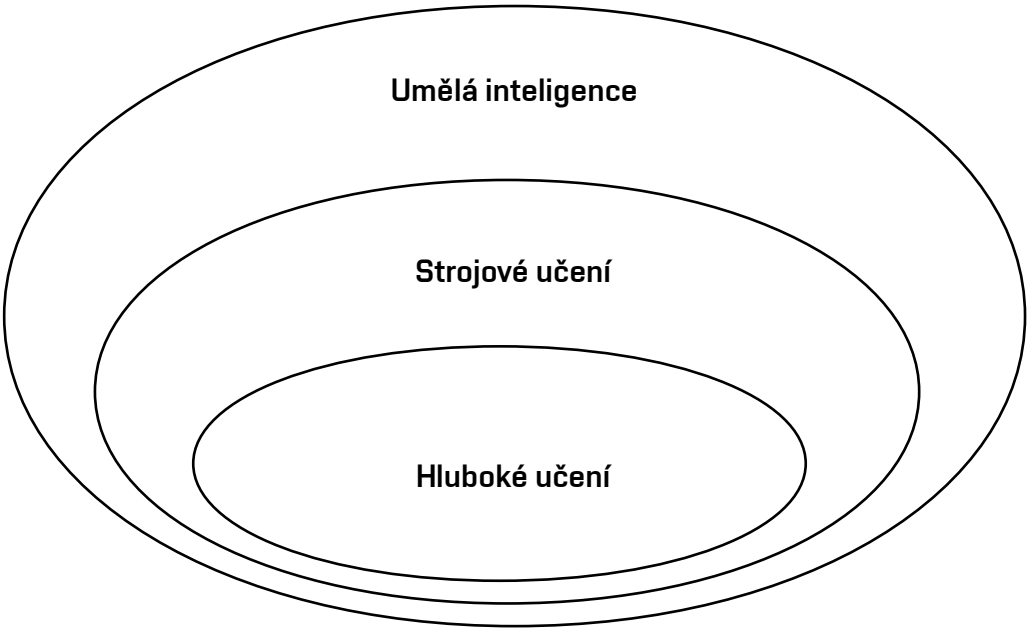
tzv. klasifikace (podrobně popsáno v kap. 5), kdy stav diagnostikovaného objektu je popsán třídou z konečné množiny tříd. K rozpoznávání můžeme například využít umělou inteligenci. Výstupem z automatizovaného diagnostického řetězce je diagnóza.[4]

Po převedení diagnostického signálu do číslicové podoby (digitalizování) jsme schopni bez omezení aplikovat metody umělé inteligence k jejich zpracování a vyhodnocení. Možné oblasti aplikace v technické diagnostice uvádí autor jako příklad (Vibrodiagnostika, Tribodiagnostika, Termodiagnostika, Elektrodiagnostika, Hluková diagnostika, Defektoskopie, Diagnostika tlaku, Diagnostika průtoku, Diagnostika výšky hladiny a další)

3 UMĚLÁ INTELIGENCE

anglicky – *Artificial Intelligence (AI)*

Za umělou inteligenci (UI) lze označit chování počítače, které částečně napodobuje lidské chování (inteligenci)[6]. Nebo [lépe – pozn. aut.] lze tvrdit, že UI je vědecká a technická disciplína, která se v současnosti velmi dynamicky vyvíjí[7], a jejíž podcenění by bylo: *„... hubris and obviously false“* - ELON MUSK[1].



Obr. 3.1: Základní součásti umělé inteligence [8]

Vzhledem k absenci celosvětově platné definice umělé inteligence, budeme UI označovat jako: vědecko-technickou disciplínu, která se zabývá různými metodami, sloužícími k řešení širokého portfólia úloh, jako například zpracování dat, rozpoznání obrazu, zvuku, oblast řešení logistických problémů, plánování, překlad textu, převod řeči na text, robotika, expertní systémy nebo konkrétněji autonomní vozidla, prediktivní údržba, doporučování obsahu, cílené reklamy ... (tento seznam by byl velmi dlouhý). Vytvořené algoritmy, programy, použité modely jsou z pravidla napsány na řešení konkrétní úlohy nebo problému,

[1]Celé tvrzení Elona Muska o tématu umělé inteligence v rozhovoru pro The New York Times z 25. 7. 2020 zní: *„My assessment about why A.I. is overlooked by very smart people is that very smart people do not think a computer can ever be as smart as they are. And this is hubris and obviously false.“* přeloženo autorem pak: „Můj názor na to, proč je UI podceňovaná chytrými lidmi je ten, že chytří lidé tvrdí, že počítač nemůže být v žádném případě tak chytrý, jako jsou oni. A to je podle mě arogantní a očividně špatně.“ Rozhovor dostupný na: `https://www.nytimes.com/2020/07/25/style/elon-musk-maureen-dowd.html`. [Online, cit. 26. 4. 2021]

k jehož řešení je vhodné použít některou z metod umělé inteligence. Metody umělé inteligence nejsou všespásné a na počátku řešení libovolného problému je si potřeba položit otázku, zda je vůbec zapotřebí aplikovat libovolnou metodu umělé inteligence. Za základní součásti UI lze považovat strojové učení (angl. *Machine Learning*) a hluboké učení (angl. *Deep Learning*), schématicky zobrazeno na obrázku 3.1.

Základní rozdíl mezi hlubokým učením a strojovým učením:[6]

- **Hluboké učení** je součástí strojového učení založeného na principu umělé neuronové sítě. Učící proces je *hluboký*, protože struktura umělé neuronové sítě je navržena tak, že se skládá z několika *vstupních*, *výstupních* a *skrytých* vrstev. Vstupní vrstvy zpracovávají data na informace, které jsou pomocí skrytých vrstev zpracovány a následně je výstupní vrstvou vytvořena predikce. Díky této struktuře se počítač sám naučí rozpoznat vstupní data a vytvořit z nich vlastní predikce.
- **Strojové učení** je součástí umělé inteligence, využívá metod (např. hlubokého učení), které umožňují počítačům využít jejich vlastních zkušeností (naučených) k řešení problému. Učící proces je založen na následujících krocích:
  1. Připravit data určená ke strojovému zpracování (již vypočtené prediktory).
  2. Naučit vytvořený model na části dat (učení).
  3. Otestovat model na zbytku dat ze souboru (testování).
  4. Uložit model s nejlepšími parametry (nastavením) pro budoucí automatickou predikci bez potřeby učit/testovat. Jinými slovy použití naučeného modelu pro predikci tříd v novém souboru dat.

3.1 Hluboké učení

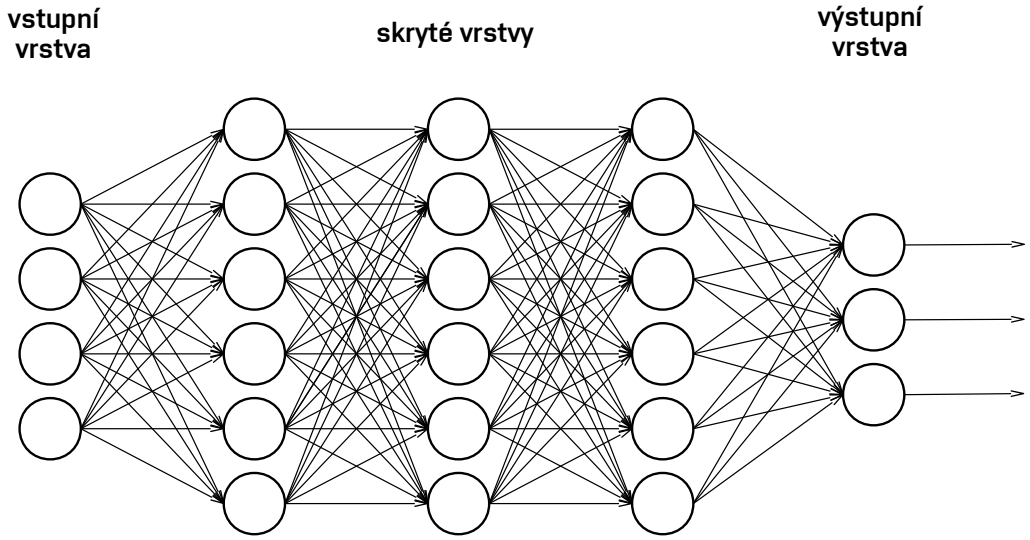
Nejprve se podívejme ještě jednou na rozdíl mezi hlubokým a strojovým učením. Pro obě metody je typické, že potřebují mnoho vstupních dat, ale jejich druh nebo spíše formát bude rozdílný. Představme si, že máme tisíce fotek zvířat a chceme vytvořit algoritmus, aplikaci, která nám pomůže odhalit koně na fotografiích.[8]

Strojové učení nemůže ze „surových“ vstupních dat najít řešení, protože vstupní data musí být nejprve označena to znamená, že musíme určit, na kterých fotografiích je kůň a na kterých nikoliv. Tomuto přístupu se říká učení s učitelem (bude popsáno podrobněji v kapitole 3.2.1. Poté již bude vytvořený algoritmus schopen odhalit fotografie na kterých se kůň vyskytuje (s určitou úspěšností). I přesto bude mít strojové učení určitá omezení nebylo by lepší, aby se vytvořený algoritmus „podíval“ na každý pixel obrázku a našel vzory, podle kterých by se rozhodl, zda „vidí“ koně? Pro strojové učení je limitem to, že musíme najít a určit vzory (charakteristické rysy) koně sami. Například tvar, kopyta, barvu, výšku – tyto charakteristiky se poté algoritmus pokusí na fotografiích najít.[8]

Co když jsou naše charakteristiky nepřesné nebo neberou v úvahu anomálie nebo výjimky? V takovém případě nám model nebude fungovat s vysokou úspěšností. Přece jen existuje mnoho plemen koní. Námi určené charakteristiky mají svůj velký mínus v tom, že neberou v potaz velké množství dat, jsou napevno dané, a nemohou se přizpůsobit,

zlepšit nebo rozrůst o další charakteristiku. To nás do jisté míry omezuje. Příkladem může být počítačový virus, který se neustále mění.[2] Pomocí hlubokého učení můžeme tento problém vyřešit. Tento algoritmus samostatně „projde“ pixel po pixelu fotografie a za pomocí neuronové sítě rozpozná vzory sám. Tento přístup napodobuje funkci lidského mozku.[8]

Schématicky je hluboké učení zobrazeno na obrázku 3.2. Vícevrstvá perceptronová síť se skládá ze vstupní vrstvy (slouží pouze jako vstup pro data, transformují je na informace), několika skrytých vrstev (s rostoucím počtem skrytých vrstev, roste i úspěšnost hlubokého učení) a výstupní vrstvy.



Obr. 3.2: Schématické zobrazení hlubokého učení pomocí neuronové sítě

Vytvoření funkční neuronové sítě pro hluboké učení vyžaduje mnohem více praktických zkušeností. Lze předpokládat, že s rostoucím objemem dat, u kterých již nebudeme schopni jednoznačně určit kategorie nebo charakteristiky, bude růst i význam hlubokého učení. Obrovskou revoluci, kterou přineslo hluboké učení popisuje lit. [9], která uvádí, že hluboké učení nám umožnilo vyvinutí dopravních prostředků bez řidiče, pomohlo zdokonalit překladač od Googlu, plynulou konverzaci se Siri nebo Alexou a přineslo obrovské zisky spojené s automatickým obchodováním na newyorské burze. Aplikace s hlubokým učením mohou hrát poker lépe než nejlepší hráči světa a také dokáže porazit člověka v deskové hře Go.

V praktické části této práce se zabýváme řešením problému pomocí strojového učení, proto mu bude věnována následující podrobná kapitola.

[2]Proto je nezbytné udržovat svůj operační systém, ale i antivirus stále aktualizovaný, respektive jeho databázi známých virů, aby je mohl včas odhalit.

3.2 Strojové učení

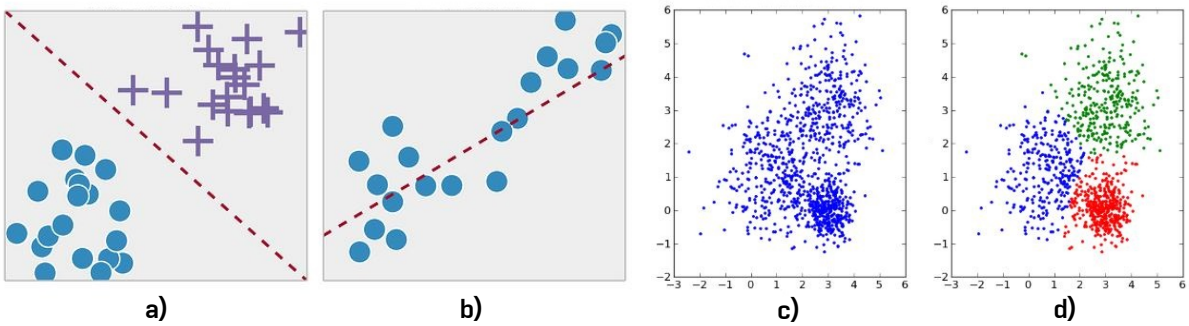
Některá algoritmy jsou jednodušší na aplikaci, a některé obsahují složitější procesy a matematické rovnice. Výhodou dnešní doby je, že již nemusíme každý algoritmus psát ručně, ale s využitím mnoha programovacích jazyků, jako například programovací jazyk *Python* [použitý v této práci – pozn. aut.], *R*, *Java*, *C*, *C++* atd., je celý proces tvorby výrazně ulehčen. Typickým ukazatelem, jak rozdělujeme různé druhy strojového učení je právě použití různých algoritmů (metod). Existují stovky různých druhů algoritmů, ale základní dělení pro strojové učení může vypadat následovně (v závorkách jsou uvedeny anglické názvy): [8]

- **Učení s učitelem** (*Supervised Learning*);
- **Učení bez učitele** (*Unsupervised Learning*);
- **Zpětnovazebné učení** (*Reinforcement Learning*);
- **Kombinace učení s učitelem a bez něj** (*Semi-Supervised Learning*).

Vzhledem k tomu, že jsou v práci aplikovány metody spadající do první kategorie (učení s učitelem), bude této kategorii věnována největší pozornost a dostatečná podrobnost pro pochopení problematiky.

Před přechodem na popis jednotlivých typů strojového učení je zapotřebí se zmínit o druzích úloh, se kterými se můžeme setkat. Mezi základní druhy úloh patří (v závorkách jsou uvedeny anglické názvy): [8],[10]

- **Klasifikace** (*Classification*) – zařazení předložených dat (prediktorů) do tříd (např. je příchozí e-mail spam nebo ne?, identifikace a rozpoznání poruchy, atd.) (obr. 3.3a)
- **Regrese** (*Regression*) – hledání závislosti v předložených datech (např. vliv systému na změnu parametru, zjištění vlivu diety na váze nebo zdravotním stavu, atd.) (obr. 3.3b)
- **Shlukování** (*Clustering*) – data jsou rozdělena na základě podobných ukazatelů – třídění do skupin shluků (obr. 3.3c,d)
- **Detekce anomálií** (*Anomaly Detection*) – zaměření se na detekci odlehlých hodnot (např. odhalování podvodů, nestandardní postupy, anomálie atd.)



Obr. 3.3: Příklady vizualizace dat pro základní úlohy: a) klasifikace, b) regrese, c) data nerozdělená do skupin (shluků), d) data rozdělená do skupin (shluků) [10]

3.2.1 Učení s učitelem

Jak již bylo zmíněno, strojové učení lze popsat jako vytvoření algoritmu, který je schopen odhalit, nebo lépe naučit se, vztahy (vazby) v datech. Pak lze říci, že učení s učitelem je podmnožinou strojového učení, které hledá vztahy mezi prediktory, které byly vypočteny a předem definovány.[11] Vzhledem k tomu, že se tato práce zabývá řešením problému klasifikace dat pomocí strojového učení a konkrétně pak za pomocí metody učení s učitelem, je vhodné uvést příklad ze života, který bude reprezentovat aplikaci učení s učitelem a to: najít vztah mezi charakteristikami domu a jeho cenou.

Očividně zde existuje nějaká vazba např. mezi počtem pokojů, obytnou plochou, nebo vzdáleností od školky, přístupností k městské hromadné dopravě atd. Hlavním úkolem učení s učitelem je tedy odhalit vztahy mezi těmito parametry, které jsme byli schopni jakýmkoliv způsobem změřit[3]. Mnoho charakteristik v domě jsme schopni převést na číselnou hodnotu (buď změřením nebo spočtením), ale máme i takové charakteristiky nebo informace, které nejsme schopni takto kvantifikovat jako například jaké sousedy budeme mít v tomto domě nebo na ulici, nebo jací poskytovatelé energií jsou dostupní v této lokaci atd. Snaha o převedení těchto vágních pojmů na čísla nám může zamezit úspěšné řešení našeho problému. Každopádně každý problém, který se snažíme vyřešit a má jednoznačně určený ukazatel, jako například cena domu, je pro nás jednoduché volit ukazatele, respektive jedno číslo, které nám jej popíše.[4][11]

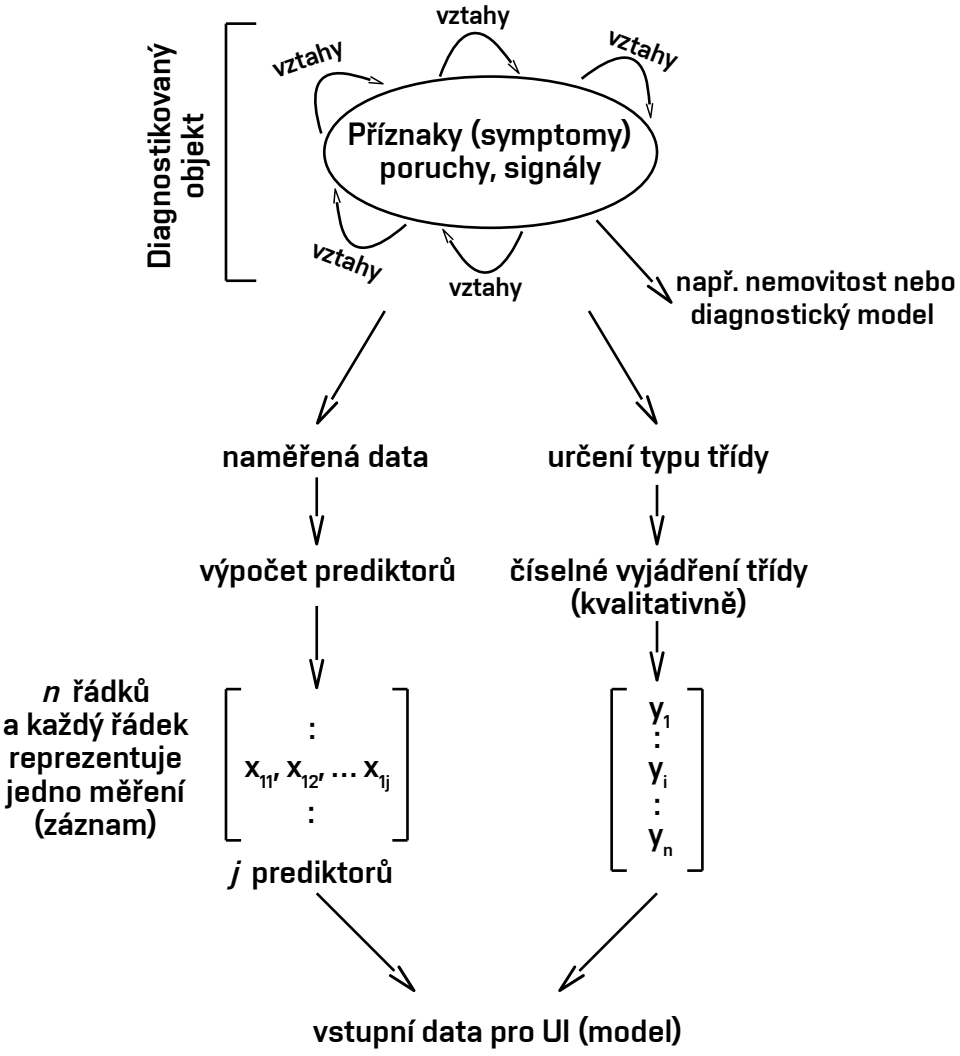
Jakmile úspěšně převedeme naměřená data do číselné podoby – charakteristik (prediktorů), musíme se rozhodnout v jaké formě tyto prediktory budeme vkládat do modelu. Jinými slovy, jakou budou mít strukturu. Možností, jak předkládat data ke zpracování máme obrovské množství, jednou z možností je předložení charakteristik jednoho domu [nebo jednoho měření např. v technické diagnostice, z diagnostického modelu – pozn. aut.] jako řádku, kterých postupně bude přibývat podle toho, kolik budeme mít změřených domů. Nyní můžeme definovat délku řádku jako počet charakteristik (prediktorů), které máme k dispozici. Z naměřených dat např. v domě [nebo diagnostického modelu, simulovaných dat – pozn. aut.] můžeme spočítat více než jednu charakteristiku (prediktor).[11] Můžeme například použít pouze změřené rozměry zahrady ke spočítání její plochy a tuto hodnotu pak použijeme jako prediktor, nebo můžeme použít naměřené rozměry základů domu a vydělit je celkovou plochou parcely, pak obdržíme prediktor, který bude vypovídat o celkové zastavěné ploše. Tak bychom mohli pokračovat dále. V kapitole 4 jsou popsána naměřená data (zrychlení v čase), ze kterých se spočítaly charakteristiky (prediktory), které nám pomohou určit, o jakou poruchu se jedná. Proces výběru, převodu nebo výpočtu dat na charakteristiky (prediktory) se nazývá extrakce charakteristických

[3]Slovem „změřit“ myslíme takové charakteristiky, které byly jasně definovány a nebo vypočítány a jsou reprezentovány číselně – kvantitativně.

[4]Obtížností v řešení netriviálních problémů, za které bychom mohli označit téměř všechny reálné problémy, je pak fakt, že ukazatel, který nám reprezentuje jasné zařazení do třídy není jediný možný. Jinými slovy, že ukazatelů je více a záleží pouze na našich zkušenostech a vědomostech, které zvolíme, respektive které mají vliv na správné zařazení to třídy.

rysů (v angličtině tento pojem hledáme pod – *feature engineering*)[11].

Učení s učitelem je tedy v podstatě určeno k nalezení vztahů mezi prediktory vypočtenými nebo určenými z naměřených dat. V praxi to znamená, že z naměřených dat spočítáme takové prediktory, které nám charakterizují hledané řešení problému – správné určení třídy. Zvolení prediktorů je do jisté míry závislé na našich zkušenostech a vědomostech, ale především je podstatné, aby zvolené prediktory dostatečně reprezentovaly hledanou třídu. Například pokud řešíme problém nalezení vztahu mezi cenou domu a počtem místností, které tento dům má, tak bychom měli model učit pomocí ceny domu jako třídy a počtem místností jako prediktoru, nebo opačně. Jinými slovy, tento naučený model nám umožní určit vztah mezi cenou domu a počtem místností – například čím více má dům místností, tím vyšší bude jeho cena. Na druhou stranu pokud je naším cílem určit cenu domu, pro který jeho cena není zveřejněna, zvolíme si cenu domu jako hledanou třídu a pomocí naučeného modelu jsme schopni predikovat očekávanou cenu této nemovitosti.[11]



Obr. 3.4: Model posloupnosti kroků v algoritmu učení s učitelem

Na obrázku 3.4 je schématicky znázorněn postup jak se dospěje od diagnostikovaného objektu přes měření diagnostického signálu, výpočtu prediktorů až k vstupním datům, která jsou předkládána modelu. Toto schéma reprezentuje algoritmus učení s učitelem.

V praxi to znamená, že modelu předložíme `n` řádků měření s námi zvoleným (vypočteným nebo určeným) počtem `j` prediktorů. Na takových datech se model „naučí“ a na další sadě dat se již modelu předloží pouze prediktory bez známého zařazení do třídy. U této sady dat již model s určitou úspěšností přiřazuje jednotlivá měření k předem definovaným třídám, které má naučené. Pokud se nám povedlo správně zvolit předkládané prediktory, které odpovídají, respektive dostatečně popisují jednotlivé hledané třídy, tak je úspěšnost přiřazení vysoká. Samozřejmě záleží na dalších faktorech, které budou podrobněji popsány u jednotlivých metod klasifikace dat v kapitole 5.

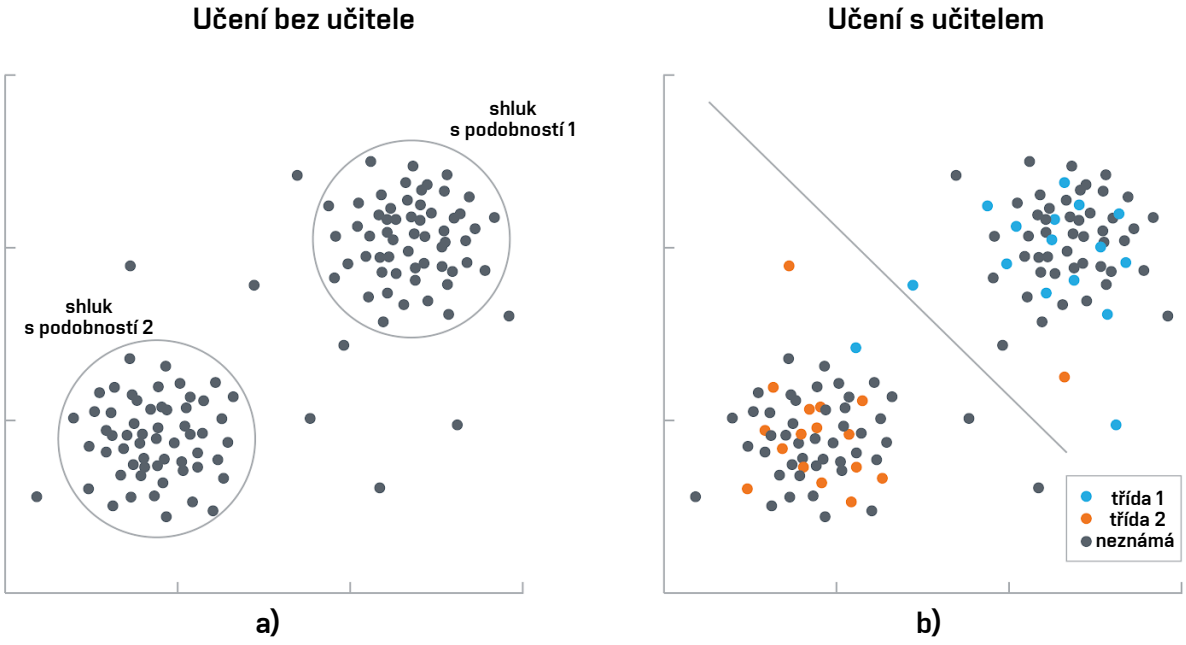
3.2.2 Učení bez učitele

Algoritmus učení bez učitele se snaží vstupní data rozdělit do tříd podle podobných znaků. Jinými slovy, pracujeme s daty, u kterých dopředu neznáme jejich třídu a výsledkem je např. shluk dat o určité podobnosti.Učení bez učitele využívá hlubokého učení k nalezení společných znaků.[8] Na obrázku 3.5 jsou znázorněny dva grafy, na který je jasně viditelné, že při učení bez učitele dojde k vytvoření několika shluků bodů, které vykazují podobné znaky. Tedy sice nevíme, o jaké třídy se jedná (jaké poruchy reprezentují), ale existují mezi nimi jisté podobnosti. Naproti tomu, když známe třídy, které se učení s učitelem snaží rozpoznat, tak dojde k rozdělení a určení hranice mezi body, které se zdají, že patří do společné třídy. Pro úplnost uveďme, že se vždy snažíme dosáhnout co nejvyšší úspěšnosti v klasifikaci, obecněji co nejlepšího výsledku. To ovšem v realitě není zdaleka tak jednoduché a proto se může stát, že se model při určování třídy splete. Více k úspěšnosti jednotlivých modelů a typům chyb v kapitole 6.

Zdaleka nejrozšířenějším přístupem při učení bez učitele je shlukování (obr.3.5a). Proces obvykle začíná odhady, následují operace a propočty vedoucí ke zlepšení výsledku. Základním úkolem je najít ta data, která mají k sobě „nejblíž“, čehož může být dosaženo pomocí různých metod například (v závorce uveden anglický název): [8]

- **Euklidovská vzdálenost** (*Euclidean Metric*) – principem je přímá vzdálenost mezi dvěma body;
- **Kosinová vzdálenost** (*Cosine Similarity Metric*) – principem je kosinus úhlu mezi dvěma vektory;
- **Manhattanská vzdálenost** (*Manhattan Metric*) – principem je nejkratší vzdálenost mezi dvěma body, jde o součet absolutních vzdáleností (např. pro dvourozměrný prostor je manhattanskou vzdáleností od bodu čtverec).

Jedna z nejčastějších aplikací algoritmu učení bez učitele je segmentace zákazníků (angl. *customer segmentation, market segmentation*), která rozděluje zákazníky do skupin (cílové skupiny) podle základních charakteristik (oblíbenost určitého produktu, preference). To umožňuje společnostem lépe plánovat marketing a cílení na zákazníka. Další



Obr. 3.5: Rozdíl mezi výsledkem algoritmů a) učení bez učitele a b) učení s učitelem [12]

velkou skupinou je tzv. analýza sentimentu (ang. *sentiment analysis*), která analyzuje textová data a snaží se určit postoj autora k danému tématu. Například může být tato analýza podstatná pro módní návrháře, z recenzí na jejich produkty lze vypozorovat co bude v nadcházejícím období módním trendem, jakým směrem se má směřovat vývoj. Existují i další úlohy pro algoritmy učení bez učitele například (v závorce je uveden anglický název):[8]

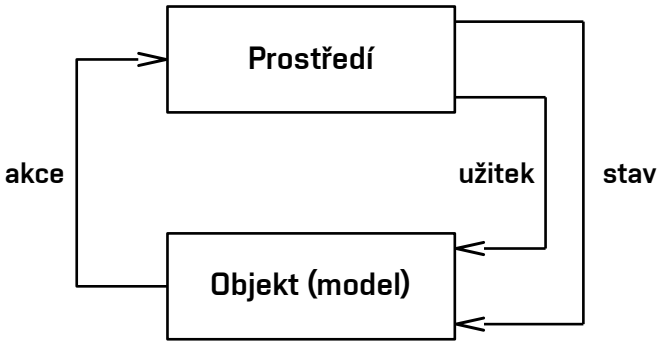
- **Asociace** (*Association*) – základní myšlenkou je: když se stane X, pak je pravděpodobné, že nastane Y, např. když si prohlížíte internetový obchod s motocykly, je pravděpodobné, že se chystáte nějaký koupit. Tento koncept může vést [a vede – pozn. aut.] k silným nástrojům doporučování obsahu, respektive podvědomého ovlivňování;
- **Detekce anomálií** (*Anomaly Detection*) – tato úloha řeší identifikaci vzdálených (odlehlých) bodů (hodnot) (ang. tzv. *outliers*) nebo jinými slovy anomálií, které mají své uplatnění např. v kyberbezpečnosti. Například odhalování podvodných e-mailů (phishing, scam, spam a další) pomocí analýzy těla zprávy, předmětu, IP adresy, odesílatele atd.
- **Autoenkodéry** (*Autoencoders*) – principem je komprese dat a následné jejich obnovení, při menším počtu neuronů dojde ke zmenšení objemu dat (jejich zjednodušení, zobecnění). Využití je například u potlačení šumu v datech.

Z popisu jednotlivých metod můžeme vycítit, že se jedná o metody, především pak asociace a detekce anomálií, které jsou v současné době nejvíce se rozvíjející, ať už z důvodu kybernetické bezpečnosti zařízení, firem nebo kritické infrastruktury státu. Asociací se bezesporu zaobírají nadnárodní společnosti jako Google nebo Facebook, s jejichž úspěšnou

nebo neúspěšnou aplikací se většina z nás setkávají takřka denně!

3.2.3 Zpětnovazebné učení

Zpětnovazebné učení jehož schématické zobrazení je na obrázku 3.6, nám říká, že se model sám učí z následků akcí, které vykonal na základě svých rozhodnutí. Ve zpětnovazebném učení existují pouze dva možné stavy (zpětné vazby) a to odměna (správně) nebo trest (špatně)[13]. Jako příklad uveďme klasické stavění nábytku z nejmenované švédské firmy. Nikdo jsme si nečetli návod k použití a tak zkoušíme jednotlivé díly skládat do sebe metodou pokus-omyl, někdy úspěšněji a někdy méně, nakonec se však dostaneme k výsledku.



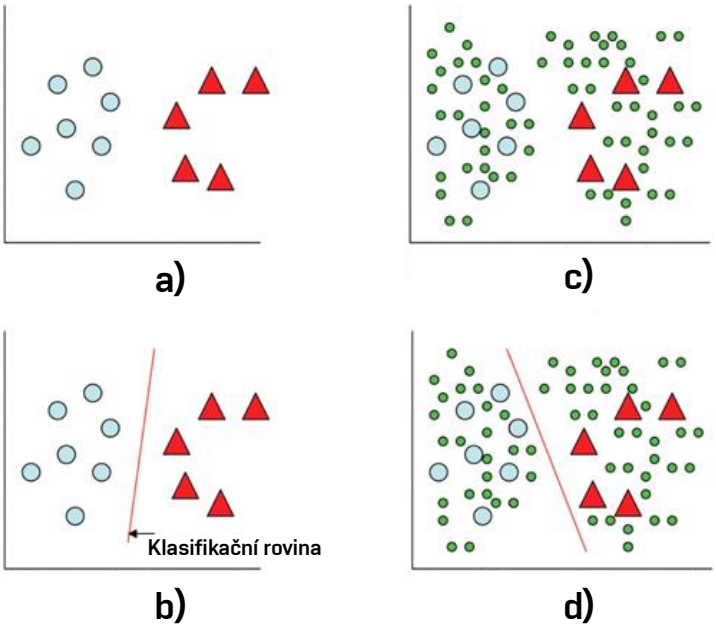
Obr. 3.6: Princip zpětnovazebného učení

Zpětnovazebné učení není zdaleka nepoužívané, jak by se mohlo na první pohled zdát. Pomocí této metody se došlo k řešení významných problému jako například:[8]

- Hry – jsou ideální oblastí aplikace zpětnovazebného učení, především protože jsou zde jasná pravidla, cíle a různá omezení (např. hrací plocha). Když vytvoříme model, můžeme provést miliony simulací, to znamená, že se systém stává velmi rychle čím dál chytřejší. Klasickým příkladem her, kde byla aplikována tato metoda jsou: šachy a čínská desková hra Go;
- Robotika – Tato metoda se stala klíčem k navigování robotů v prostoru. Pokud budeme mít například roboticky vysavač, tak právě jeho kontakty s okolním prostředím ho učí, respektive navigují skrze celou místnost.

3.2.4 Kombinace učení s učitelem a bez něj

Jak již z názvu vyplývá, jde o kombinaci dvou metod, které již byly popsány. Tato metoda se využije, když máme k dispozici malé množství dat u kterých známe třídu (učení s učitelem) a velké množství dat u kterých je třída neznámá (učení bez učitele). V některých případech jsou rozsáhlé datové soubory bez určených tříd velmi složité na identifikaci s nízkou úspěšností. S výhodou se proto může využít tohoto algoritmu. Na obrázku 3.7c, d je zobrazen princip určení třídy pro nezařazená data, kdy jejich třída bude určena podle jejich vzdálenosti (nejkratší) k jednomu z bodů, u kterého známe zařazení do třídy. [14]



Obr. 3.7: Rozdíl mezi algoritmy učení s učitelem a učení bez učitele: a) data se známou třídou, b) učení s učitelem, c) částečně data se známou třídou, ale převažuje počet dat s třídou neznámou, d) kombinace učení s učitelem a bez něj [14]

Jako příklad z reálného života může být výuka matematiky, respektive řešení algebraických problémů. Dětem se vysvětlí základní algebraické operace (sčítání, násobení, ...), předvede se pár příkladů u kterých znají výsledek (a postup). Během života pak takovýchto algebraických operací budou řešit mnoho a bez předem známého postupu nebo řešení. Vzhledem k tomu, že kdysi měli k dispozici „návod“ k řešení podobných úloh, je pro ně řešení nových problému mnohem snazší a úspěšnost se výrazně zvýší.

Dalším zajímavým využitím tohoto přístupu může být například magnetická rezonance, kde radiolog označí část těla, která se má skenovat a poté pomocí hlubokého učení počítač rozpozná souvislosti, vzorce (angl. – *patterns*) v tomto obraze a stanový výstup.[8]

3.3 SW prostředky a knihovny

Ačkoliv nám VUT v Brně poskytuje bezplatný přístup[5]k programovému prostředí MATLAB, ve kterém by bylo samozřejmě možné řešit zadaný problém, tak je tato práce zaměřena na programovací jazyk Python a jeho knihovny. Součástí rešerše je i seznam dalších softwarů, které lze využít k řešení problému z oblasti zpracování dat za pomocí umělé inteligence (strojového učení). Programovacímu jazyku Python bude věnována největší pozornost a některé použité knihovny budou podrobněji popsány v kapitole 3.3.2.

Programovací prostředí, ve kterém byla práce řešena se jmenuje Spyder[6], jedná se o volně dostupné vědecké prostředí napsané v jazyce Python a navržené pro vědce, inženýry nebo jakékoliv datové analytiky. Obsahuje kombinaci pokročilých funkcí pro úpravy, analýzu, úpravu a komplexní vývojové nástroje s možností vyhledávání v datech, interaktivním zobrazení, podrobnou kontrolou a přehlednou vizualizací výsledků. Editor Spyder je například součástí volně dostupné platformy Anaconda[7], která obsahuje stovky knihoven, poznámek nebo projektů a především prostředí pro všestranné využití.

3.3.1 SW pro práci s umělou inteligencí

Existuje spousta volně dostupných softwarů (ale i placených) pro práci s umělou inteligencí. Níže jsou uvedeny a stručně popsány některé z nich.

TensorFlow

TensorFlow[8] je volně dostupná bezplatná platforma pro strojové učení. Má komplexní a flexibilní nástroje, knihovny a rozšířenou komunitu, která umožňuje výzkumným pracovníkům prosazovat nejmodernější řešení v oblasti strojového učení. Umožňuje vývojářům snadno vytvářet a nasazovat aplikace založené na strojovém učení. Mezi společnosti, které využívají TensorFlow patří Airbnb, CocaCola, Google, Intel, Twitter a další.[15]

PyTorch

PyTorch[9] je volně dostupná bezplatná platforma určená pro strojové učení, založená na vědecké výpočetní knihovně Torch. Umožňuje rychlé experimentování a efektivní psaní kódu v uživatelsky příjemném prostředí s množstvím použitelných nástrojů a knihoven. PyTorch podporuje cloudové služby, jednoduchou práci s předpřipravenými modely a zapojení GPU pro spuštění modelů s velkým objemem dat. Mezi společnosti, které využívají PyTorch patří Salesforce, Stanford University, Udacity a další.[16]

[5]Licence byla pořízena začátkem roku 2020 viz. `https://www.vutbr.cz/vut/f19528/d194063`, přístup k softwaru jsme jako studenti nebo pracovníci VUT měli i předtím, ale pouze na školních PC.

[6]`https://www.spyder-ide.org/`

[7]`https://anaconda.org/`

[8]`https://www.tensorflow.org/`

[9]`https://pytorch.org/`

Keras

Keras[10] (celým názvem v angličtině *Keras: the Python deep learning API*) vývojáři definují jako aplikační programovací rozhraní (API) navržené pro lidi, nikoliv stroje. Základní koncepcí je dodržení osvědčených postupů pro snižování kognitivní zátěže tj. nabízí konzistentní a jednoduché rozhraní, minimalizuje počet akcí vyžadovaných pro uživatele při běžném používání a poskytuje jasné chybové zprávy. Obsahuje rozsáhlou dokumentaci a příručky pro vývojáře. Keras je založen TensorFlow 2.0, využívá jejich největších předností a je plně kompatibilní s platformou TensorFlow. Specializuje se na hluboké učení. Mezi společnosti, které Keras využívají patří CERN konrétně pak LHC (Velký hadronový urychlovač angl. *Large Hadron Collider*), NASA, NIH (Národní ústav zdraví pro Spojené státy americké, angl. *National Institute of Health*) a další. Keras je volně dostupný a bezplatný.[17]

KNIME

KNIME[11] je volně dostupný a bezplatný software určený pro datovou analýzu v jednoduchém interaktivním prostředí. Software obsahuje nástroje pro dolování z dat (ang. *Data Mining*) ale také strojového učení. KNIME využívá modulární koncepci toku dat programátorem vytvořenou posloupností. Jinými slovy, že si sami sestavíme posloupnost bloků s jasně danými vstupy a výstupy. Výstup z jednoho elementu tvoří vstup pro následující. Software obsahuje i přehlednou vizualizaci dat. Mezi společnosti, které KNIME používají patří Siemens, Continental, T-Mobile a další.[18]

GNU Octave

Octave[12] je jedna z dalších možností, ve které se můžeme věnovat strojovému učení, jedná se o matematicky orientovaný, programovací jazyk. Umožňuje práci jak s vektory, tak maticemi. Software, který využívá tento jazyk je volně dostupný a bezplatný. Navíc je tento jazyk kompatibilní s jazykem Matlabu (jehož software je placený).[19]

MATLAB

MATLAB[13] patří mezi zástupce placených softwarů, ale autor považuje za nezbytné se o něm stručně zmínit. Jedná se o programovací platformu vyvinutou především pro výzkumné a technické prostředí určené pro analýzu dat. Jeho programovací jazyk má shodné jméno jako software a je určený pro práci s maticovým zápisem. Aplikace MATLABu umožňují práci pokrývající celou umělou inteligenci (strojové učení, hluboké učení atd.). Jsou designovány jako „klikací“ aplikace pro trénink a porovnávání vytvořených modelů. Umožňuje rozsáhlou analýzu dat pro rozpoznání charakteristických rysů (angl. *feature*

[10] `https://keras.io/`
[11] `https://www.knime.com/`
[12] `https://www.gnu.org/software/octave/`
[13] `https://www.mathworks.com/`

extraction) a automatickou volbou nejlepšího nastavení parametrů modelu (angl. *hyper-parameter tuning*).[20, 21] Celé prostředí je uživatelsky velmi příjemné a studenti FSI VUT v Brně se v něm učí rozvíjet základy programování a psání kódů. Navíc je nyní k dispozici MATLAB Online, díky kterému již nepotřebujeme zdlouhavou instalaci kompletního softwaru do našich PC. Jednoduše z pohodlí jakéhokoliv zařízení s připojením k internetu můžeme pomocí internetového prohlížeče pracovat.

V další kapitole bude popsán programovací jazyk Python, ve kterém byla zhotovena praktická část této práce a knihovny, které byly použity k řešení problematiky klasifikace dat ve strojovém učení.

3.3.2 Programovací jazyk Python a použité knihovny

V jednoduchosti je síla, tak by se dal jednoduše popsat programovací jazyk Python[14]. Tento jazyk se považuje za standard ve vývoji a tvorbě umělé inteligence. Především pak právě kvůli jeho jednoduchosti, všestrannosti a využitelnosti. Navíc vzhledem k bezplatnému přístupu a možnosti vytváření různých přídavných modulů nyní obsahuje stovky různých knihoven a balíčků, které se soustředí na umělou inteligenci.

Výčet základních knihoven, které byly použity v této práci.

`NumPy`

`NumPy` neboli numerický Python tvoří jednu ze základních knihoven programovacího jazyka Python, která slouží k různým matematickým výpočtům. Jedním z přínosů této knihovny jsou multidimenzionální objekty typu pole, nad kterými je možné rychle a efektivně vykonávat různé matematické nebo logické operace, třídění, výběr, diskrétní Fourierovu transformaci, základní lineární algebru a mnoho další. Základ této knihovny tvoří tzv. `ndarray`. Tento objekt obsahuje n-dimenzionální pole homogenních dat a různé operace, které mohou být nad nimi vykonány. Rozdíly od ostatních sekvencí v Pythonu např. `list` nebo `tuple` jsou:[22]

- Velikost `ndarray` je dána pevně již při založení, na rozdíl od klasického `listu`, který se může dynamicky měnit i po založení. Změnou velikosti `ndarray` dojde k založení nového a starý bude smazán.
- Od běžných sekvencí se liší také tím, že veškeré prvky `ndarray` musí být stejného typu.
- `Numpy` usnadňuje práci při aplikaci pokročilých matematických operací s velkým množství dat. Zpravidla se takové operace vykonávají efektivněji a s menším množstvím kódu k zapsání, než pomocí integrovaných sekvencí.
- Rostoucí množství vědeckých a matematických balíčků založených v jazyku Python používá pole `NumPy`. I když tyto balíčky obvykle podporují vstup v základních sekvencích Pythonu, převádějí takový vstup na pole `NumPy` před zpracováním a často vydávají pole `NumPy` jako výstup. Jinými slovy, aby bylo možné efektivně využívat většinu dnešního vědeckého-matematického softwaru založeného na Pythonu,

[14]Podle Guida van Rossuma (autor filozofie) byl Python vytvořen s jasnou představou že:

- krása je lepší než ošklivost;
- explicitní je lepší než implicitní;
- jednoduchý je lepší než složitý;
- složitý je lepší než komplikovaný;
- plochý je lepší než vnořený;
- řídký je lepší než hustý;
- srozumitelnost (čitelnost) se počítá.
- a další

Zen of Python citováno z: `https://www.python.org/dev/peps/pep-0020/` [online, cit. 1. 5. 2021]

nestačí jen vědět, jak používat vestavěné typy sekvencí Pythonu - je také potřeba vědět, jak používat pole `NumPy`.

Pandas

`Pandas` je další volně dostupnou knihovou pro programovací jazyk Python. Je určená pro analýzu dat, které jsou reprezentovány v 2D tabulce (jako práce s tabulkou v Excelu, ale mnohem automatizovanější). Příkladem těchto dat jsou SQL tabulky, CSV soubory a nebo různé tabulkové procesory. Základní datové typy, se kterými tato knihovna pracuje jsou:[23]

- `DataFrame` – tento datový typ je identický v práci s tabulkou v Excelu, má řádky (angl. *rows*) a sloupce (angl. *columns*).
- `Series` – reprezentuje sloupce tabulky, které obsahují libovolné hodnoty jako seznam, ale mají navíc datový typ a „index“, které jednotlivé hodnoty pojmenovávají. S informacemi ve sloupcích se dá libovolně počítat (základní aritmetické operace) nad každou hodnotou ve sloupci.

V práci byla tato knihovna například použita k načtení sady dat v .txt formátu na datový typ `DataFrame` příkazem:

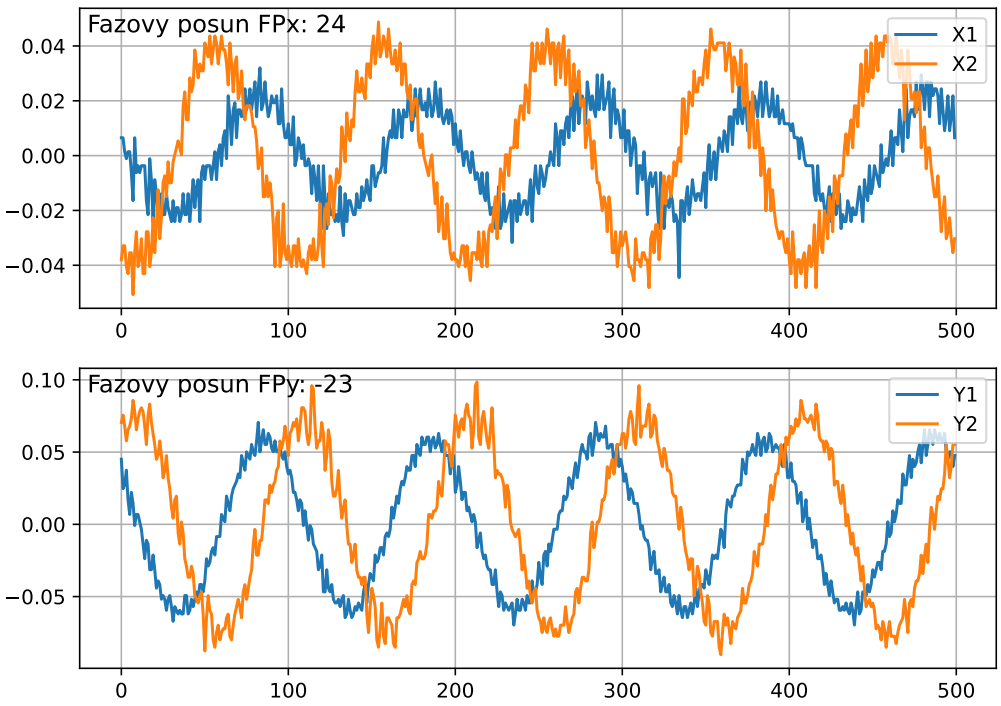
```
improt pandas as pd
dataset = pd.read_csv("data_uceni.txt", sep = '\t')
```

Matplotlib

Tato knihovna je určena pro vytváření 2D grafů. Základní filozofií autora (John D. Hunter) bylo umožnit jednoduché vytvoření grafu za použití minimálního počtu příkazů (pokud možno jedním). Pokud si potřebujeme datový soubor graficky zobrazit nebo vytvořit grafický výstup, neměli bychom trávit mnoho času voláním dalších příkazů nebo nastavováním vlastností objektu.[24] Příklad části použitého kódu pro vizualizaci naměřených dat z obrázku 4.4 z kapitoly 4.2, konkrétně pro OK stav osy `y`:

```
plt.subplot(5,2,2)
plt.plot(n,Y1,label="pozice 1")
plt.plot(n,Y2,label="pozice 2")
plt.grid()
plt.legend(loc="upper right")
plt.title('OK osa y')
plt.xlim(0,0.5)
plt.ylim(-0.1, 0.1)
```

Další příklad může být rychlá vizualizace části dat pro ujištění, že psaný kód je „správně funkční“. Jinými slovy pro účely ladění se zpracuje pouze jedno měření a zobrazí se v obrázku 3.8.



Obr. 3.8: Ladění kódu, zobrazení jednoho měření pro fázový posun stojanu 1, 2 obou os

Příklad zápisu kódu pro tento obrázek 3.8:

```
plt.clf()
n=500
plt.subplot(2,1,1)
plt.plot(A1[0:n],label="X1")
plt.plot(B1[0:n],label="X2")
plt.grid()
plt.legend(loc="upper right")
plt.title(f'Fazovy posun FPx:{FPx}',loc='left',y=1.0,pad=-12)


plt.subplot(2,1,2)
plt.plot(A2[0:n],label="Y1")
plt.plot(B2[0:n],label="Y2")
plt.grid()
plt.legend(loc="upper right")
plt.title(f'Fazovy posun FPy:{FPy}',loc='left',y=1.0,pad=-12)
```

SciPy

Název vědecký Python SciPy (angl. *Scientific Python*) vystihuje samotnou podstatu této použité knihovny. Obsahuje moduly určené pro optimalizaci, lineární algebru, interpolaci, zpracovaní obrazu a a další. Jaký je tedy rozdíl mezi knihovnami Numpy a SciPy? Knihovna NumPy a její základní objekt ndarray tvoří samotný základ pro knihovnu SciPy a její další funkce. Lze tety tvrdit, že knihovna SciPy je její nádstavbou a spolu s využitím dalších knihoven jako například zmiňované Pandas nebo Matplotlib tvoří SciPy základní podporu jazyka Python pro vědecké a výpočetní úlohy.[25]

Scikit-learn

Prakticky nejdůležitějším nástrojem, pomocí kterého byla tato práce vypracována je právě scikit-learn. Tato knihovna obsahuje všechny aplikované algoritmy učení s učitelem které byly použity v této práci (a další, také pro učení bez učitele). Je založená na základech SciPy – tato knihovna musí být nainstalována před použitím algoritmů ze samotné knihovny scikit-learn. Nezahrnuje nástroje pro načítání, manipulaci ani sumarizaci dat, soustředí se především na modelování z dat. Pro předchozí operace je zapotřebí využít jiných knihoven, jako například NumPy nebo Pandas.[26]

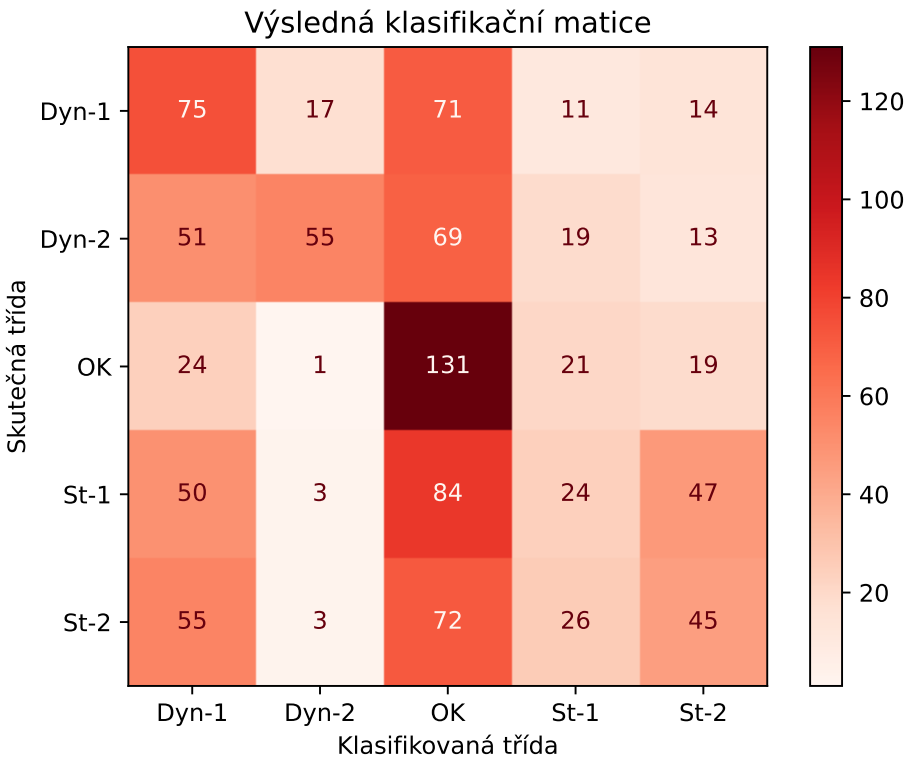
Každý z použitých modelů je reprezentovaný jako třída, která využívá metodu fit(), která trénuje model pomocí klasifikátorů a metodu predict(), která aplikuje naučený model na testovací set dat[25].

Modely, které byly použity v této práci z knihovny scikit-learn neboli sklearn jsou:

- z balíčku neighbors metoda KNeighborsClassifier viz. kap. 5.3
- z balíčku svm metoda SVC viz. kap. 5.4
- z balíčku tree metoda DecisionTreeClassifier viz. kap. 5.5
- z balíčku ensemble metoda RandomForestClassifier viz. kap. 5.6

Po dokončení klasifikace je potřeba zjistit úspěšnost našeho modelu. K tomu mohou sloužit implementované metody z balíčku metrics v knihovně sklearn, mezi které patří confusion.matrix a accuracy.score. Každopádně pro vhodnější interpretaci je výslednou klasifikační matici (angl. *confusion matrix*) vhodné ještě upravit, respektive převést do grafické podoby. Pro vizualizaci výsledné klasifikační matice byl použit balíček plot_confusion_matrix[15] z knihovny sklearn.metrics. Příklad vizualizace klasifikační matice je na obrázku 3.9. Jak se v ní orientovat a vyčíst z ní důležité informace o výsledku klasifikace, budou popsány v následujícím odstavci. Pro úplnost uveďme, že tato klasifikační matice nemá nic společného s výsledky dosaženými v praktické části této práce (kromě shodných názvů klasifikovaných tříd OK, ST-1, ...). Slouží pouze jako demonstrační jednotka pro popis informací, které obsahuje.

[15]Tento balíček je podrobně popsán na: https://scikit-learn.org/stable/modules/generated/sklearn.metrics.plot_confusion_matrix.html



Obr. 3.9: Ukázková klasifikační matice

Řádky reprezentují skutečnou třídu vzorků, které byly modelu předloženy pro klasifikaci. Součet všech hodnot v řádcích dává počet celkových vzorků, předložených pro klasifikaci. V tomto případě jich bylo předloženo přesně 1000.

- 188 z třídy Dyn-1
- 207 z třídy Dyn-2
- 196 z třídy OK
- 208 z třídy St-1
- 201 z třídy St-2

Sloupce jsou pak modelem klasifikovaná třída pro předložené vzorky. Důležité informace v sobě nese hlavní diagonála[16]. Co je pro nás velmi podstatné, je správná klasifikace OK stavu (třídy). Tedy na třetím řádku (OK) jsme předložili celkem 196 vzorků, model úspěšně klasifikoval 131 z nich. To znamená, že úspěšnost klasifikace OK stavu je podíl mezi správně klasifikovanými vzorky z OK třídy a celkovým počtem předložených vzorků z OK třídy 131/196 je 66,84 % úspěšnost – to není nikterak uspokojující. Tímto způsobem můžeme projít celou matici. Druhý podstatný údaj, který zde na první pohled není bohužel vidět, je celková úspěšnost klasifikace, lze využít `accuracy.score` pro její výpočet,

[16]Hlavní diagonála libovolné matice je posloupnost tvořená prvním prvkem z prvního řádku, druhým prvkem z druhého řádku atd. Hlavní diagonála matice tedy vede z levého horního rohu do pravého dolního rohu.

nebo si dát do poměru součet správně klasifikovaných tříd (součet na hlavní diagonále) ku celkovému počtu předložených vzorků. 330/1000 je 33,00 % úspěšnost – opět to není dobrý výsledek.

Obecně lze tvrdit, že se snažíme dosáhnout 100 % klasifikace OK stavu, jinými slovy, aby byl model (umělá inteligence) schopen rozeznat bezporuchový stav (OK) od stavu poruchového. To, s jakou úspěšností bude rozeznávat druhy poruchových stavů mezi sebou, je druhořadé. Ale v některých aplikacích může být tento parametr také velmi důležitý a sledovaný. Pro celkovou úspěšnost klasifikace se snažíme dosáhnout minimálně 90 %, vše co je nad tuto hodnotu považujeme za dobrý výsledek.

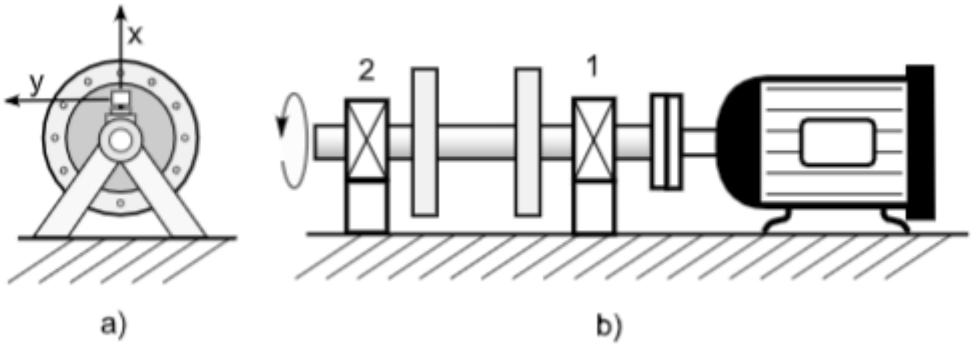
Více o zlepšování úspěšnosti klasifikace, prediktorech a aplikovaných metodách bude probráno v kapitole 5.

4 KLASIFIKOVANÁ DATA

Jak bylo popsáno v kapitole 2.2 diagnostický signál lze získat v různých oblastech technické diagnostiky. V této práci byla použita naměřená data z modelu na simulování poruchy rotačního stroje – oblast vibrodiagnostiky. Pro úplnost uveďme, že naměřená data byla analyzována a matematicky zpracována v časové oblasti[1].

4.1 Vibrodiagnostický model

Model byl vytvořen pro simulaci poruchy na rotačním stroji. Schéma modelu je na obr. 4.1.



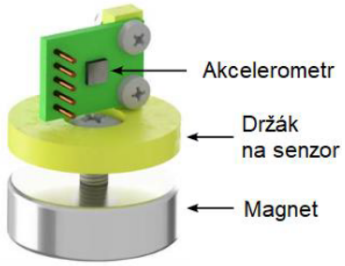
Obr. 4.1: Diagnostický model: a) pohled zleva a směry os, b) pohled zepředu a znázorněné pozice snímačů [27]

Naměřená data jsou pak získávána ze dvou MEMS (Micro Electro Mechanical Systems) akcelerometrů ADXL335[2], které jsou umístěny na stojanech ložisek (pozice 1 a 2) na obr. 4.1. Jedná se o analogové tříosé akcelerometry, jejich výstupní signál je zpracován měřicí kartou NI USB-6009[3]s nastavením vzorkovací frekvence na 2 kHz a 1000 zaznamenávanými vzorky. Akcelerometr je přišroubován k plastovému držáku dvěma šrouby, tento plastový držák je pak jedním šroubem uchycen k magnetické základně, schématicky zobrazeno na obr. 4.2. Tato soustava je pak připevněna ke stojanům ložisek.[27]

[1]Analýza dat v časové oblasti není jedinou oblastí, kterou můžeme analyzovat. Další možností by byla např. analýza ve frekvenční oblasti. Vzhledem k tomu, že se v této práci zaměřujeme na klasifikaci dat, kdy je model učen s učitelem (více o umělé inteligenci a jejích možnostech v kapitole 3) a jde nám o správnou identifikaci nějaké formy nevývahy (poruchy), je volba analýzy dat v časové oblasti vhodnou volbou. Více k analýze a zpracování diagnostických signálů v lit. [5].

[2]Specifikace akcelerometru ADXL335 je dostupná na: `https://www.analog.com/en/products/adxl335.html`. [Online, cit. 18. 4. 2021]

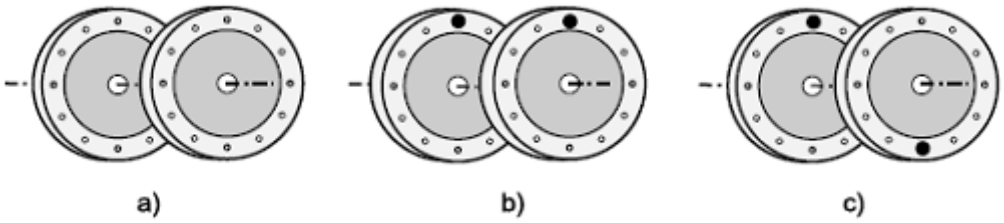
[3]Specifikace měřicí karty NI USB-6009 je dostupná na: `https://www.ni.com/cs-cz/support/model.usb-6009.html`. [Online, cit. 18. 4. 2021]



Obr. 4.2: Soustava pro uchycení akcelerometru [27]

4.2 Simulované poruchy

Takto vytvořený model (obr. 4.1) byl pro potřebu simulování poruchy ještě osazen dvěma kotouči s otvory po obvodové části, schéma těchto kotoučů na obr. 4.3. Do obvodových děr byla následně umístěna závaží pro simulaci dvou typů poruchového stavu a to statické nevývahy a dynamické nevývahy. Rozdíl v simulování statické a dynamické nevývahy spočívá v poloze umístění závaží v obvodu kotouče, kdy pro statickou nevývahu jsou závaží umístěna ve stejné poloze na obou kotoučích, a pro dynamickou nevývahu je poloha jednoho závaží posunuta o 180° vůči prvnímu. Byly použity dvě sady závaží pro simulaci dvou úrovní nevývah. 2,5 gramová závaží pro první úroveň a 3,7 gramová závaží pro úroveň druhou.[27]



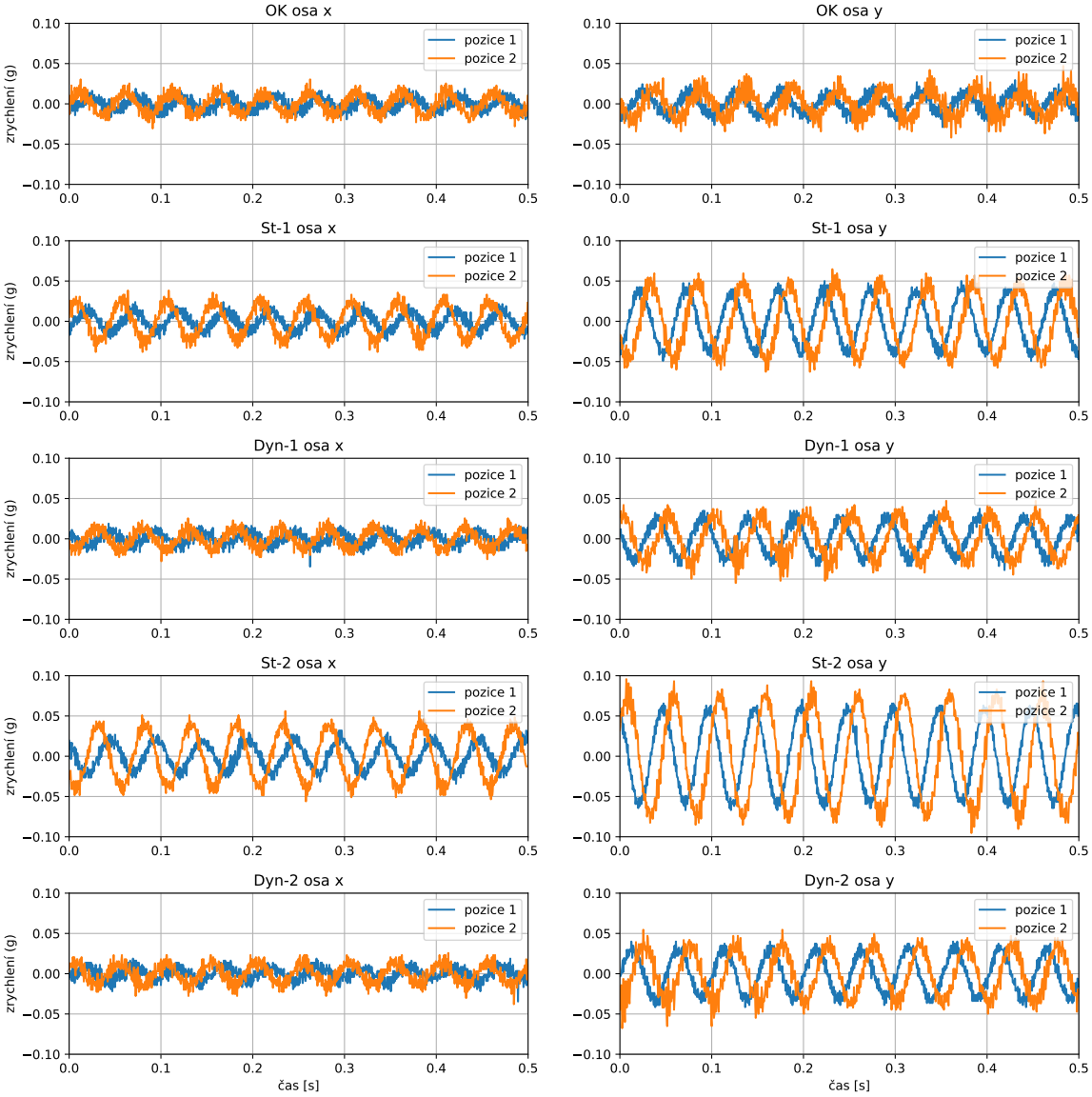
Obr. 4.3: Schéma umístění závaží na kotouči: a) bezporuchový stav (bez závaží), b) statická nevývaha, c) dynamická nevývaha [27]

Bylo provedeno pět různých měření. Konkrétně pak bezporuchový stav, statická nevývaha první a druhé úrovně (rozdílné sady závaží), a také dynamická nevývaha opět první a druhé úrovně (rozdílné sady závaží). Tato měření byla označena za stavy, v jakém se diagnostický model může nacházet. Byly pojmenovány:[27]

- **OK** – Bezporuchový stav, kotouče nejsou osazeny závažím (obr. 4.3a)
- **St-1** – Statická nevývaha první úrovně, kotouče jsou osazeny sadou závaží o hmotnosti 2,5 g (obr. 4.3b)
- **Dyn-1** – Dynamická nevývaha první úrovně, kotouče jsou osazeny sadou závaží o hmotnosti 2,5 g (obr. 4.3c)

- **St-2** – Statická nevývaha druhé úrovně, kotouče jsou osazeny sadou závaží o hmotnosti 3,7 g (obr. 4.3b)
- **Dyn-2** – Dynamická nevývaha druhé úrovně, kotouče jsou osazeny sadou závaží o hmotnosti 3,7 g (obr. 4.3c)

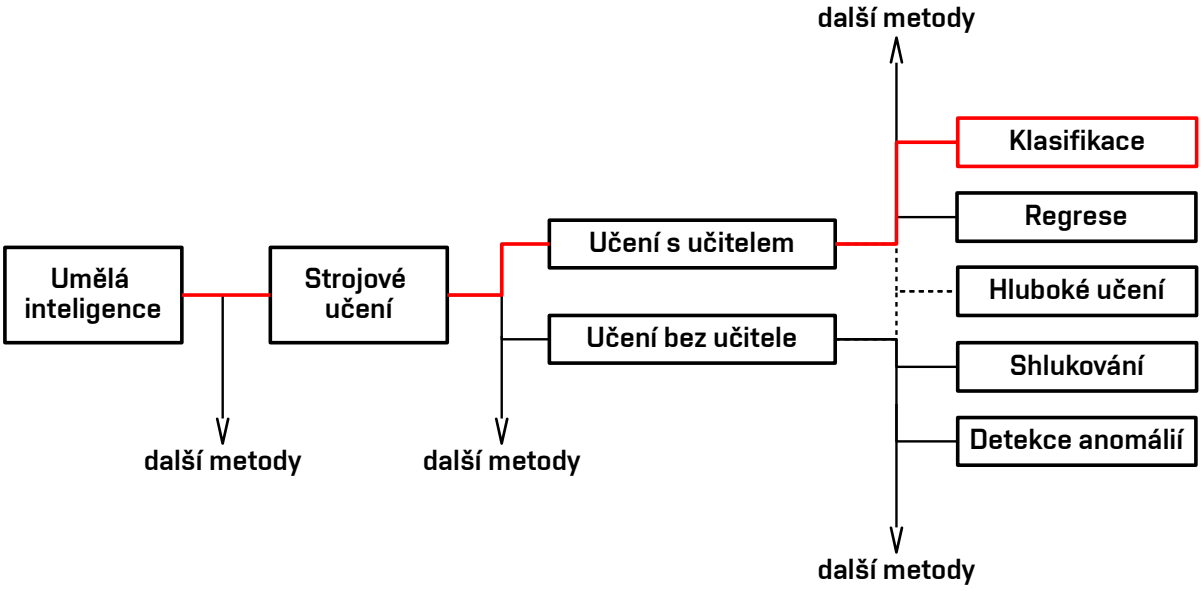
Při jednoduché vizualizaci naměřených dat a vynesení jednoho řádku záznamu do grafu pro každý typ poruchy (5) a každou osu (x a y) dostaneme obrázek 4.4.



Obr. 4.4: Vizualizace naměřených dat pro jednotlivé poruchy a měřené osy

5 APLIKOVANÉ METODY KLASIFIKACE DAT

Před popisem samotných metod se ještě podívejme na přesné zařazení klasifikace dat do širokého portfolia vědecko-technické disciplíny – umělá inteligence. Na obrázku 5.1 je červeně vyznačená cesta od disciplíny ke klasifikaci dat. Pro metodu učení s učitelem[1] je typické, že prediktory, díky kterým se model naučí rozeznávat jednotlivé třídy (druhy) poruch, musíme (jakožto učitel) nejprve určit (vypočítat) z naměřených dat.



Obr. 5.1: Zařazení klasifikace dat do umělé inteligence

5.1 Prediktory

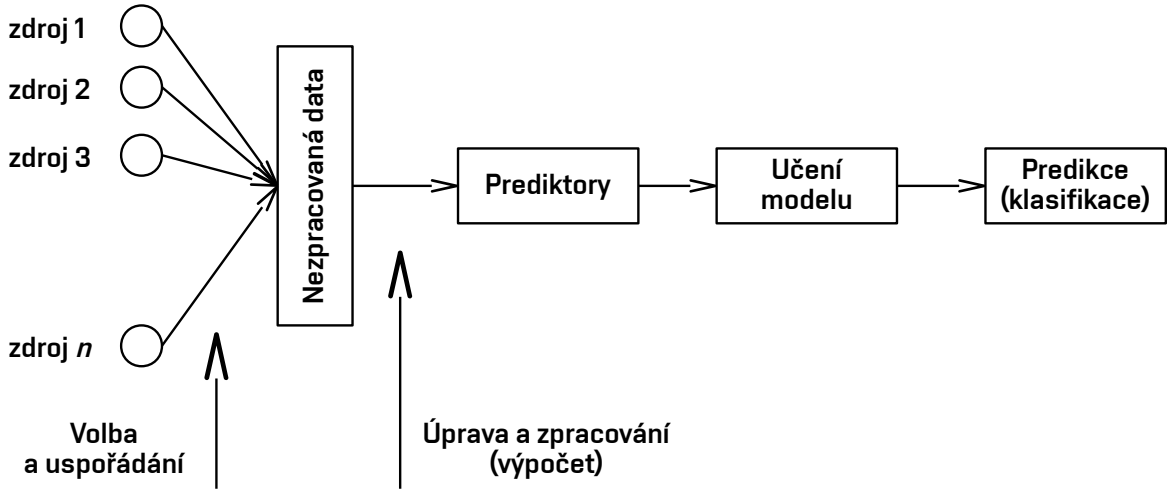
Procesu extrakce charakteristických rysů (prediktorů) se věnuje celá literatura, např. [28] nebo podrobné kapitoly 4.1 a 4.2 v lit. [5]. Autor by neváhal označit tento proces jako jednu samostatnou část umělé inteligence – přece jen to není triviální úkon, musíte znát dobře svůj diagnostický objekt, abyste jej uměli dobře popsat pomocí prediktorů. Jinak vám vytvořený model pro klasifikaci dat nemusí dávat dobrý výsledek – nutně to nemusí být chyba modelu.

Za prediktor můžeme označit číselné reprezentování nezpracovaných dat. V praxi to znamená, že máme mnoho možností jak z naměřených dat zpracovat (vypočítat) prediktory, proto záleží do jisté míry i na našich zkušenostech a vědomostech. Podstatné je, že prediktory musí vycházet z naměřených dat, která máme k dispozici. Jinými slovy, dokáží vhodně popsat objekt diagnostiky. Odlišné problémy si žádají různá řešení, některé

[1] To je hlavní rozdíl od metody učení bez učitele, respektive hlubokého učení, kde si neuronová síť za pomocí skrytých vrstev „vypočítá prediktory sama“ a následně se podle nich rozhodne.

modely je vhodné popsat prediktory vypočtenými z průběhu časové oblasti, některé z frekvenční oblasti atd. Prediktor by měl, v ideálním případě, co nejvíce vystihovat hledaný problém (poruchu) a pro model by měl být jednoduchý na zpracování.[28]

Počet prediktorů hraje také významnou roli. Pokud není k dispozici dostatek relevantních prediktorů, dostáváme se zpět na začátek problému, že model nebude schopen poskytnout dobrý výsledek. Na druhou stranu, když budeme mít příliš mnoho prediktorů, nebo když většina z nich nebude podstatná, tak může být obtížné model „naučit“. Při procesu učení by se mohlo něco pokazit a model by byl tak nepoužitelný.[28] Navíc bychom prodlužovali dobu zpracování (výpočtu) prediktorů a také učení modelu – to by také znamenalo, že by se prodloužila doba určení predikce, která by s největší pravděpodobností neměla vysokou úspěšnost. Zařazení procesu výpočtu prediktorů do strojového učení je na obrázku 5.2.



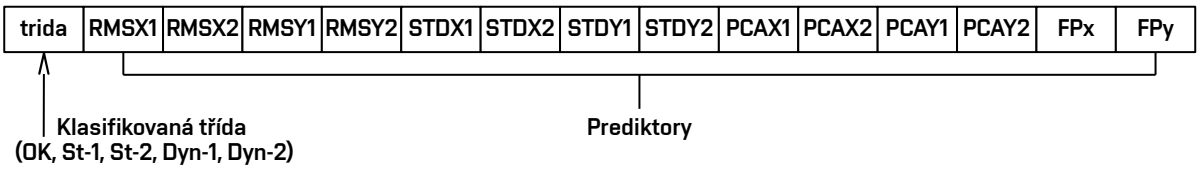
Obr. 5.2: Zařazení výpočtu prediktorů do procesu strojového učení [28]

Dále jsou stručně popsány zvolené prediktory, které byly požity v této práci. Mají za úkol jasně charakterizovat simulované poruchy (statická a dynamická nevývaha ve dvou úrovních):

- **RMS** – efektivní hodnota (angl. – *root mean square*), tento prediktor má vypovídat o mohutnosti vibrací.
- **STD** – výběrová směrodatná odchylka (ang. – *standard deviation*), laicky řečeno nám tento prediktor vypovídá o tom, jak moc je diagnostikovaný objekt nevyvážený.
- **PCA** – analýza hlavních komponent (angl. – *principal component analysis*), je statistická metoda na hledání korelací, často slouží ke snížení dimenze dat.
- **FP** – fázový posun (v angličtině hledat pod názvem – *cross correlation*), pomáhá rozlišit mezi statickou a dynamickou nevývahou (posun mezi dvěma signály).

Výsledný soubor, který se předkládá modelu pro učení nebo pro klasifikaci, má schématicky podobu na obrázku 5.3. Obsahuje v prvním sloupci informaci o poruše (`St-1, St-2, Dyn-1, Dyn-2`) nebo bezvadném stavu (`OK`) a 14 prediktorů. Prediktory RMS,

STD a PCA jsou vypočítány pro každý stojan a každou osu, výjimku tvoří prediktor FP, který je pouze pro osy `x` a `y` – vychází to z předpokladu pro fázový posun, že při nevývaze dojde k posunu signálu mezi stojanem 1 a stojanem 2.



Obr. 5.3: Formát předkládaných prediktorů pro klasifikační model

5.2 Hledání nejvhodnějších parametrů modelu

anglicky – *Hyperparameter Tuning*

Každý klasifikační model, který bude popsán v nadcházejících kapitolách, má vlastní nastavení „nějakých“ parametrů, které mohou do jisté míry ovlivnit úspěšnost výsledné klasifikace nebo její rychlost provedení. Například u metody podpůrných vektorů (SVM) můžeme zvolit jaký tvar má mít křivka, která bude rozdělovat klasifikované třídy (přímka, křivka různého stupně (polynomu) `n` atd.). Funkce, kterou knihovna `sklearn` nabízí se jmenuje `GridSearchCV`. Příklad zápisu v aplikaci:

```
from sklearn.model_selection import GridSearchCV
param_grid = (kombinace parametrů, které chci
                vyzkoušet u použité metody)
grid = GridSearchCV(estimator=(použitá metoda),
                param_grid=(prohledávané parametry), ...)
grid.fit(X, y)
>>>
Vypíše nejvhodnější kombinaci parametrů
```

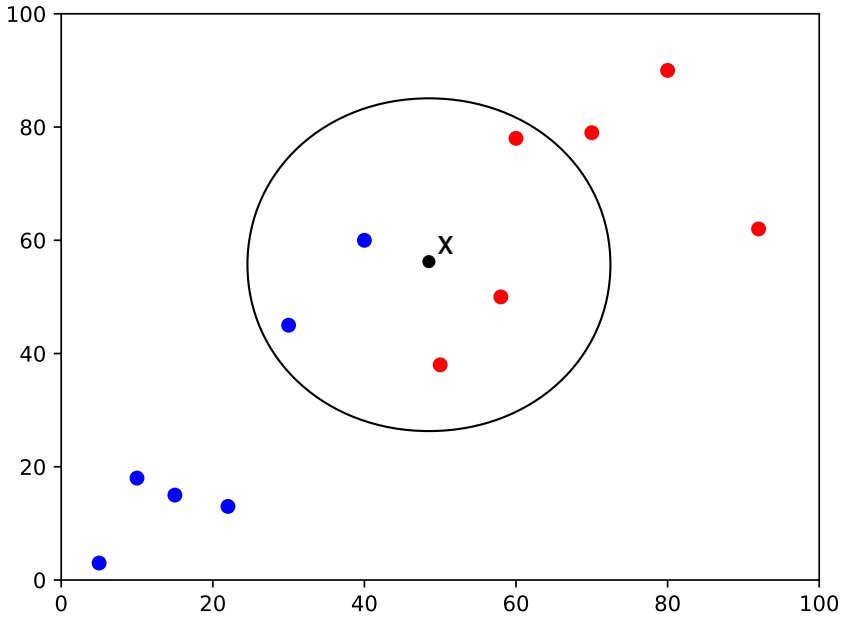
Druhou možností, kterou knihovna `sklearn` nabízí je `RandomizedSearchCV`, rozdíl oproti `GridSearchCV` spočívá v tom, že neprochází všechny možné kombinace parametrů, které požadujeme vyzkoušet. S rostoucím počtem parametrů (nebo jejich rozsahem) se prodlužuje i výpočetní doba. V této práci si můžeme dovolit projít všechny kombinace parametrů, doba výpočtu není dlouhá.

5.3 Klasifikace metodou nejbližšího souseda

anglicky – *K-Nearest Neighbors (KNN)*

Metoda klasifikace nejbližšího souseda (k-nejbližších sousedů) se dá označit za statistickou metodu, protože je založena na porovnávání vzdáleností, od posuzovaného bodu (vzorku) k-nejbližších sousedů. Při klasifikaci pak dochází k výběru `k` nejbližších sousedů od posuzovaného bodu. Posuzovaný bod je zařazen do takové třídy, jehož třída má nejvyšší výskyt zastoupení u `k` nejbližších sousedů. Zvolením vysoké hodnoty parametru `k` snižuje pravděpodobnost, že bude klasifikace nepřiměřeně ovlivněna okolním šumem. Avšak, na druhou stranu, volbou vysoké hodnoty parametru `k` dochází ke snížení přesnosti klasifikace.[29]

Například neklasifikovaný bod `X = [50,58]` na obr. 5.4 pro jehož model platí, že se řídí parametrem `k = 5` (těchto pět nejbližších „sousedů“ bylo ohraničeno kruhem) bude výsledná klasifikovaná třída „červená“, protože z pěti nejbližších bodů právě tři náleží do červené třídy[2].



Obr. 5.4: Příklad určení třídy metodou KNN

Mezi výhody patří:[30]

- metoda je jednoduchá a snadno se aplikuje;
- odpadá potřeba náročnějšího učení modelu například oproti metodě SVM (někdy se metodě KNN také říká „líný algoritmus učení“);
- metoda má všestranné využití, může být použita ke klasifikaci, regresi nebo hledání.

[2]Takto jednoduchou klasifikaci si můžeme dovolit graficky zobrazit – jedná se pouze o dvoudimenzionální prostor.

Nevýhodami jsou:[30]

- metoda KNN není vhodná pro data, jejichž vzorky mají velké množství klasifikátorů, protože s velkým množstvím klasifikátorů narůstá i obtížnost výpočtu vzdálenosti pro každou dimenzi;
- pokud mají předložená data velké množství vzorků, tak se prodlužuje doba výpočtu, protože se musí spočítat vzdálenost všech vzorků od klasifikovaného bodu;
- tuto metodu nelze dobře aplikovat na kategoriální data (kvalitativní znaky[3]– hodnoty jsou neporovnatelné, nelze je seřadit, příkladem jsou krevní skupiny, národní příslušnost atd.).

Pro metodu k-nejbližších sousedů využijeme z knihovny `sklearn.neighbors` klasifikátor `KNeighborsClassifier`[4]. Pro tento model byly určeny tyto nejvhodnější parametry:

```
KNeighborsClassifier(n_neighbors=1, weights='uniform',
                     algorithm='ball_tree')
```

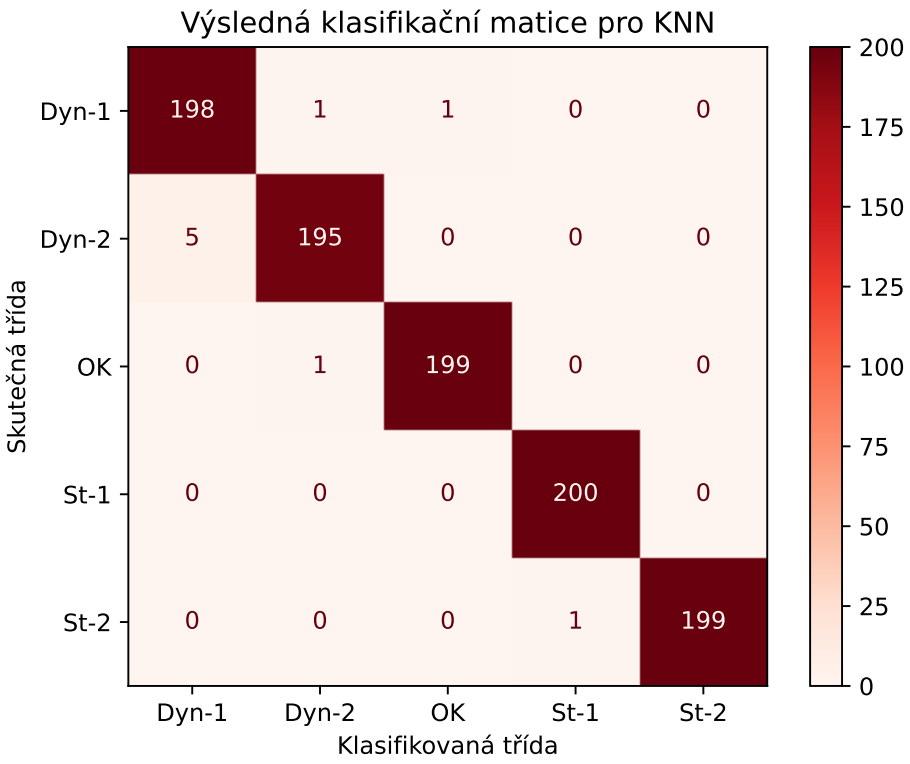
Celkem bylo desetkrát testováno učení dat, jehož smyslem bylo statisticky určit přesnost při učení, výsledky jsou zaznamenány v tabulce 5.1. Bylo dosaženo 100,00±0,00 % úspěšnosti při učení.

Tab. 5.1: Záznam testování učení modelu KNN

učení	**přesnost**
1	1,000
2	1,000
3	1,000
4	1,000
5	1,000
6	1,000
7	1,000
8	1,000
9	1,000
10	1,000

[3]Mezi kvalitativní znaky také patří udělované hodnocení ve škole.

[4]Tento klasifikátor je podrobně popsán na: `https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html`



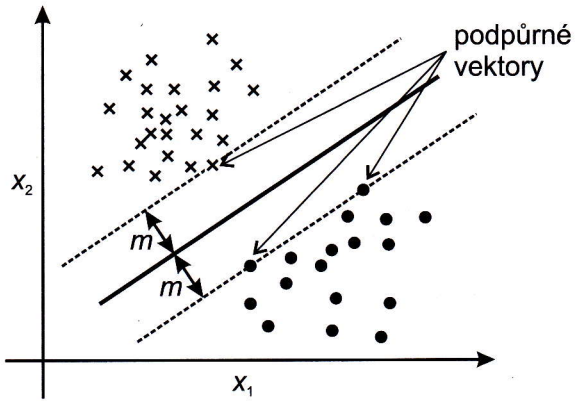
Obr. 5.5: Výsledná klasifikační matice pro KNN

Na obrázku 5.5 je zobrazena výsledná klasifikační matice pro metodu nejbližšího souseda. Testovány byly vzorky dat, které model nikdy „neviděl“, tj. rozdílný datový soubor, než na kterém byl model „učen“. Model úspěšně klasifikoval 991 vzorků z 1000 předložených, což je celková úspěšnost klasifikace 99,10 %. Z 200 předložených vzorků pro OK stav dokázal správně klasifikovat 199 z nich, jeden vzorek chybně určil jako dynamická nevývaha druhé úrovně tj. úspěšnost 99,5 %. Tato hodnota, zda je model schopen správně rozlišit mezi OK stavem a poruchovým, je pro nás výrazně důležitější, než samotná klasifikace přiřazení poruchového stavu k jeho příčině.

5.4 Metoda podpůrných vektorů

anglicky – *Support Vector Machines (SVM)*

Metoda podpůrných vektorů je založena na principu minimalizace strukturálního risku. Tento problém je převeden na hledání maximální vzdálenosti rozdělující roviny klasifikátoru k bodům z předkládané množiny – těmto vzdálenostem se říká okraje, na obr. 5.6 označeny jako `m`. Následně řešení hledání maximální vzdálenosti převede na problém nalezení minimálního kvadratického kritéria. Zde můžeme výhodně použít to, že ve vhodném vyjádření se vektor trénovací množiny vyskytuje ve formě skalárních součinů, což nám umožní použít jádrovou funkci (angl. *kernel function*). Jádrová funkce definuje tvar nelineární rozdělující nadplochy tj. nelineární klasifikátor.[5]



Obr. 5.6: Optimální rozdělující přímka (tlustě) – lineární klasifikace SVM [5]

Mezi výhody patří:[31]

- velká efektivita pro data s mnoha prediktory;
- vysoká účinnost (lze bez problému použít) i pro datové soubory, které mají vyšší počet prediktorů než vzorků;
- využívá podmnožinu tréninkových vzorků v rozhodovací funkci (podpůrné vektory), díky tomu je model robustně naučen;
- všestranná aplikace díky možnosti výběru jádrové funkce, která definuje tvar „nadplochy“.

Nevýhodami jsou:[31]

- pokud je počet prediktorů výrazně vyšší než počet vzorků, je potřeba dát pozor na tzv. přeučení (angl. *over-fitting* – kdy bychom se mohli snažit model příliš moc specifikovat, v takovém případě je vhodnější se zaměřit na vhodný výběr prediktorů;
- pro některá použití metoda SVM přímo neposkytuje odhady pravděpodobnosti, ty je nutné získat pomocí křížové validace.

Pro metodu podpůrných vektorů využijeme z knihovny `sklearn.svm` klasifikátor `SVC`[5]. Pro tento model byly určeny tyto nejvhodnější parametry:

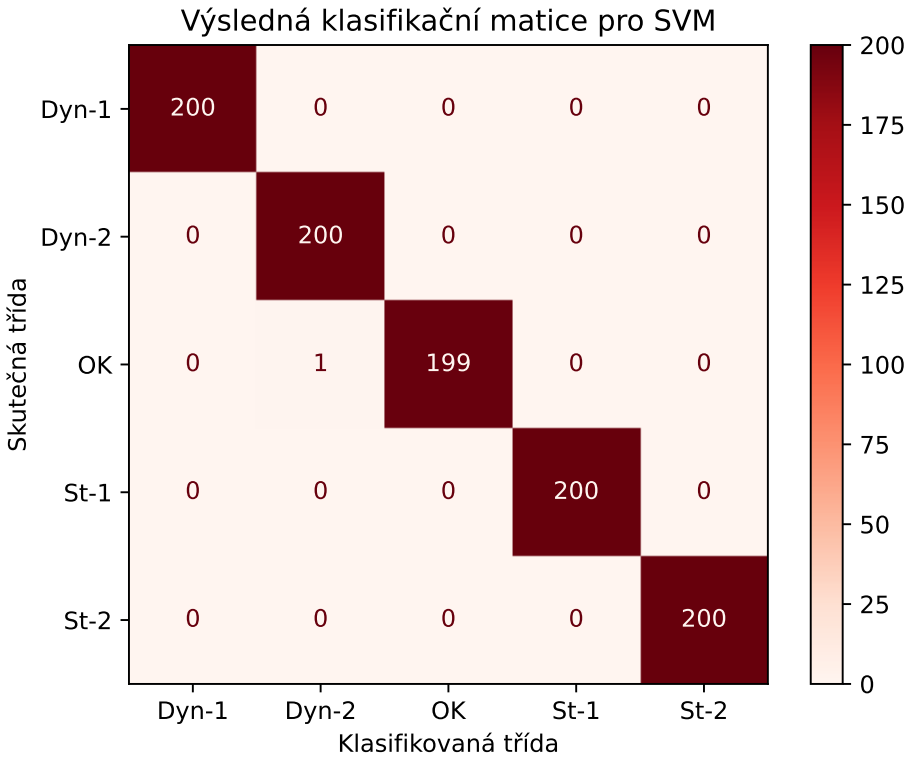
```
SVC(C=200, kernel='rbf')
```

Celkem bylo desetkrát testováno učení dat, jehož smyslem bylo statisticky určit přesnost při učení, výsledky jsou zaznamenány v tabulce 5.2. Bylo dosaženo 100,00±0,00 % úspěšnosti při učení.

Tab. 5.2: Záznam testování učení modelu SVM

učení	**přesnost**
1	1,000
2	1,000
3	1,000
4	1,000
5	1,000
6	1,000
7	1,000
8	1,000
9	1,000
10	1,000

[5]Tento klasifikátor je podrobně popsán na: `https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html`



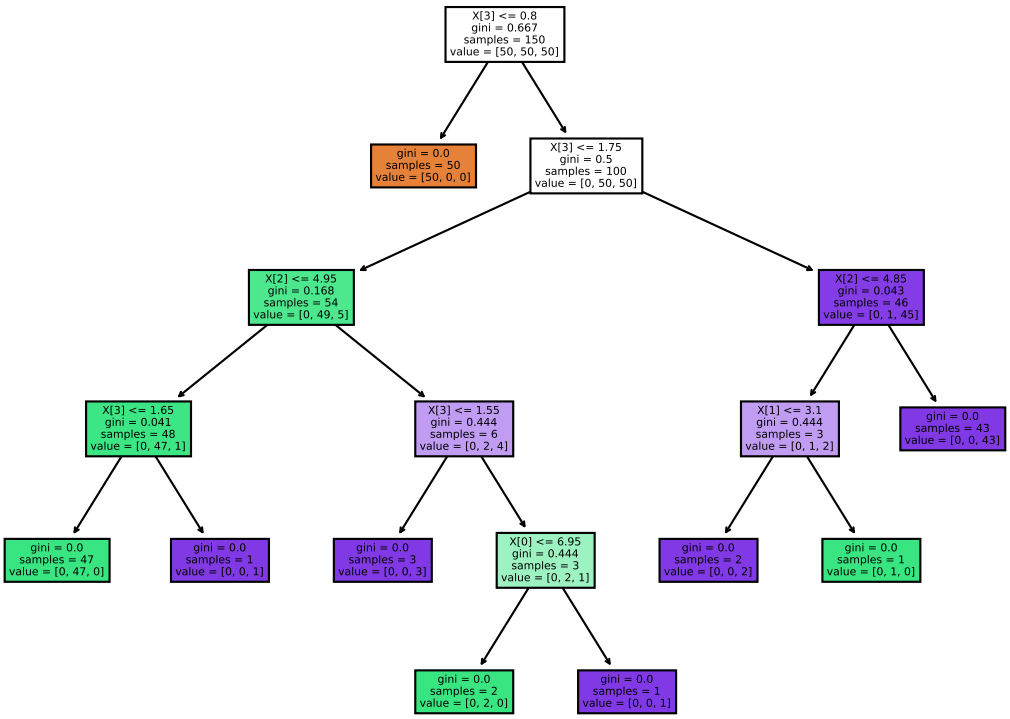
Obr. 5.7: Výsledná klasifikační matice pro SVM

Na obrázku 5.7 je zobrazena výsledná klasifikační matice pro metodu podpůrných vektorů. Testovány byly vzorky dat, které model nikdy „neviděl“, tj. rozdílný datový soubor, než na kterém byl model „učen“. Model úspěšně klasifikoval 999 vzorků z 1000 předložených, což je celková úspěšnost klasifikace 99,90 %. Z 200 předložených vzorků pro OK stav dokázal správně klasifikovat 199 z nich, jeden vzorek chybně určil jako dynamická nevývaha druhé úrovně tj. úspěšnost 99,5 %. Tato hodnota, zda je model schopen správně rozlišit mezi OK stavem a poruchovým, je pro nás výrazně důležitější, než samotná klasifikace přiřazení poruchového stavu k jeho příčině.

5.5 Rozhodovací stromy

anglicky – *Decision Trees (DT)*

Metoda rozhodovacích stromů je jednou z velmi rozsáhle používaných a jednoduchých metod, kterou lze aplikovat zejména na problémy regrese a klasifikace dat. I když je myšlenka rozhodovacích stromů[6]relativně jednoduchá, jedná se o efektivní nástroj umělé inteligence. Pro každý prediktor je vytvořený algoritmus rozhodnutí v rámci rozhodovacího stromu. Ty jejichž váha je nejvyšší, respektive dokáží nejlépe rozdělit jednotlivé třídy, jsou umístěny u vrcholu stromu. Prvnímu rozhodnutí se také někdy říká kořenový uzel, od něhož postupujeme skrze rozhodnutí nebo podmínky až ke koncovému uzlu (listu), který reprezentuje zařazení do třídy pro posuzovaný vzorek. Schématicky je tato posloupnost zobrazena na obr. 5.8.[32] Na obrázku jsou klasifikované celkem tři třídy (barevně rozlišené koncové uzly – oranžová, zelená, fialová) a existují různé způsoby jak se k nim dostat. V kořenovém uzlu je rozhodnuto zda posuzovaný vzorek patří do oranžové třídy, či nikoliv. Pokud do ní nenáleží, postupuje se dále a rozhoduje se již pouze mezi zelenou a fialovou třídou. U každého dalšího rozhodnutí má model strukturovaně připravené metriky rozhodnutí, podle kterých se bude řídit. Koncové uzly (bez dalšího dělení) jsou klasifikované třídy pro posuzovaný vzorek.



Obr. 5.8: Grafické zobrazení rozhodovacího stromu pro 3 třídy [33]

[6]Rozhodovací strom můžeme označit za orientovaný graf s definovaným vstupem – rozhodovacími uzly – konečnými akcemi (klasifikací).

Mezi výhody patří:[34]

- metoda je jednoduchá na pochopení a interpretaci, lze graficky zobrazit (nejsme omezeni počtem dimenzí – prediktorů);
- malé nároky na předzpracování vstupních dat;
- možné ověření modelu statistickými testy tzn. lze určit spolehlivost modelu;
- metoda je robustní proti vnějším rušivým vlivům, působících na model (šum).

Nevýhodami jsou:[34]

- automatické vytvoření rozhodovacího stromu, bez zásahu programátora, může vést k přeučení, takto vytvořený model by byl příliš komplikovaný a nedosahoval by požadované úspěšnosti v klasifikaci;
- problémem rozhodovacích stromů je, že dovedou rozhodnout v uzlech o lokálně optimálních rozhodnutích, nedovedou zaručit globálně nejvhodnější rozhodnutí;
- některé rozhodovací stromy jsou velmi nevyvážené, protože váha některých vstupních parametrů je příliš vysoká, to může negativně ovlivnit proces rozhodování a klasifikaci.

Pro metodu rozhodovacích stromů byla využita knihovna `sklearn.tree` a klasifikátor `DecisionTreeClassifier`[7]. Pro tento model byly určeny tyto nejvhodnější parametry:

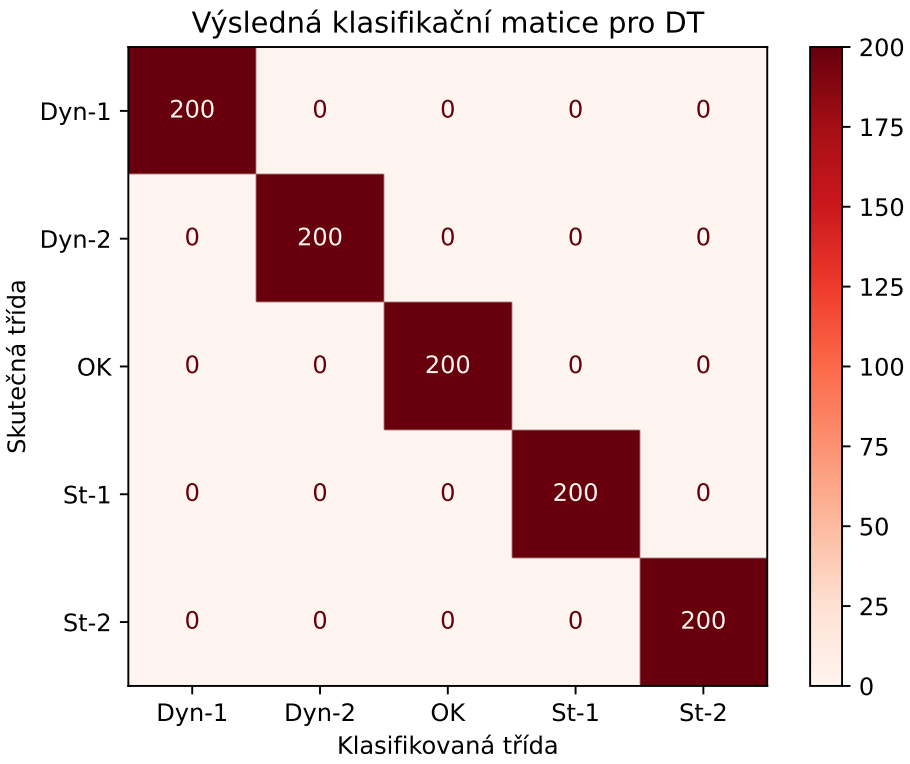
```
DecisionTreeClassifier(criterion='gini', min_samples_leaf=1,
                       splitter='best')
```

Celkem bylo desetkrát testováno učení dat, jehož smyslem bylo statisticky určit přesnost při učení, výsledky jsou zaznamenány v tabulce 5.3. Bylo dosaženo 100,00±0,00 % úspěšnosti při učení.

Tab. 5.3: Záznam testování učení modelu DT

učení	**přesnost**
1	1,000
2	1,000
3	1,000
4	1,000
5	1,000
6	1,000
7	1,000
8	1,000
9	1,000
10	1,000

[7]Tento klasifikátor je podrobně popsán na: `https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html`



Obr. 5.9: Výsledná klasifikační matice pro DT

Na obrázku 5.9 je zobrazena výsledná klasifikační matice pro model rozhodovacího stromu. Testovány byly vzorky dat, které model nikdy „neviděl“, tj. rozdílný datový soubor, než na kterém byl model „učen“. Všem klasifikovaným vzorkům model správně přiřadil jejich skutečnou třídu. Tento model má úspěšnost klasifikace 100 %. Umí rozpoznat OK stav od poruchového. V rámci poruchového stavu dále umí úspěšně rozpoznat o jakou poruchu se jedná (statická nebo dynamická i úroveň jedna nebo dvě).

5.6 Klasifikace metodou náhodného lesa

anglicky – *Random Forests Classifiers (RF)*

Jak již anglický název napovídá, metoda náhodného lesa obsahuje velké množství jednotlivých rozhodovacích stromů, které dohromady tvoří funkční celek. Tato metoda bere výhody rozhodovacích stromů a přidává k nim ještě něco navíc (zlepšuje je). Trénovací soubory pro jednotlivé stromy jsou bootstrapové[8]výběry z datového souboru. Při volbě větvení pro daný uzel se z celkového počtu prediktorů náhodně zvolí jen některé a dále již větvení probíhá stejným způsobem jako bylo popsáno v kap.5.5. Takováto náhodná volba prediktorů urychluje dobu výpočtu, ale také zvyšuje celkovou úspěšnost klasifikace. Lze tvrdit, že nezáleží na výběru prediktorů – tento výběr může být zcela náhodný![35]

Jinými slovy kombinace vytvořených rozhodovacích stromů z náhodně zvolených prediktorů, které jsou mezi sebou relativně nekorelované (nemají vzájemný vztah), dohromady vytvoří jeden model (náhodný les), který výrazně překoná jakýkoliv dílčí samostatný rozhodovací strom (z pohledu úspěšnosti klasifikace). Důvodem, proč tento model dosahuje vyšší úspěšnosti v klasifikaci je ten, že jednotlivé stromy se navzájem „chrání“ před individuálními chybami (za předpokladu, že chyba není ve všech).[36]

Mezi výhody patří:[37]

- díky náhodnému výběru prediktorů je každý strom trénován na podmnožině dat a tak si navzájem pomáhají najít globálně nejvhodnější rozhodnutí – spoléhají na sílu „davu“;
- metoda náhodného lesa je velmi stabilní i když se model setká s daty, které nikdy neviděl, nebo jsou odlišné od těch, na kterých se učil, důvodem je, že k ovlivnění dojde u jednotlivých rozhodovacích stromů, nikoliv však u celého lesa;
- oproti rozhodovacím stromů metoda náhodného lesa dosahuje velmi dobrých výsledků jak u numerických klasifikátorů, tak u kategoriálních;
- náhodný les je plně funkční i když ve vstupních datech jsou chybějící hodnoty nebo pokud nejsou vstupní data více zpracována (škálována).

Nevýhodami jsou:[37]

- problém u metody náhodného lesa může nastat při délce výpočtu, tato metoda je poměrně složitá a výrazně náročnější na výpočetní výkon než rozhodovací stromy (samostatně);
- z důvodů vyšší složitosti může být proces učení výrazně delší u velkého množství vstupních dat (klasifikovaných tříd a prediktorů).

[8] *„**Bagging** je akronym, zkratka z angl. „**b**ootstrap **agg**regat**ing**“, která z trénovacího datového souboru vytvoří náhodným výběrem s vracením L souborů velikosti n (bootstrapových výběrů), každý z nich se použije k sestrojení jednoho klasifikačního stromu a výsledný klasifikační les je pak dán většinovým hlasováním se stejnými vahami (resp. aritmetickým průměrem dílčích regresních funkcí).*“ Více k této metodě a hlubší podstatě fungovaní nejen metody náhodného lesa v lit. [35], ze které bylo citováno.

Vyjmenované nevýhody u aplikace v této klasifikační práci nebyly zaznamenány. Pracuje se s relativně malým datovým souborem.

Pro metodu náhodného lesa byla využita knihovna `sklearn.ensemble` a klasifikátor `RandomForestClassifier`[9]. Pro tento model byly určeny tyto nejvhodnější parametry:

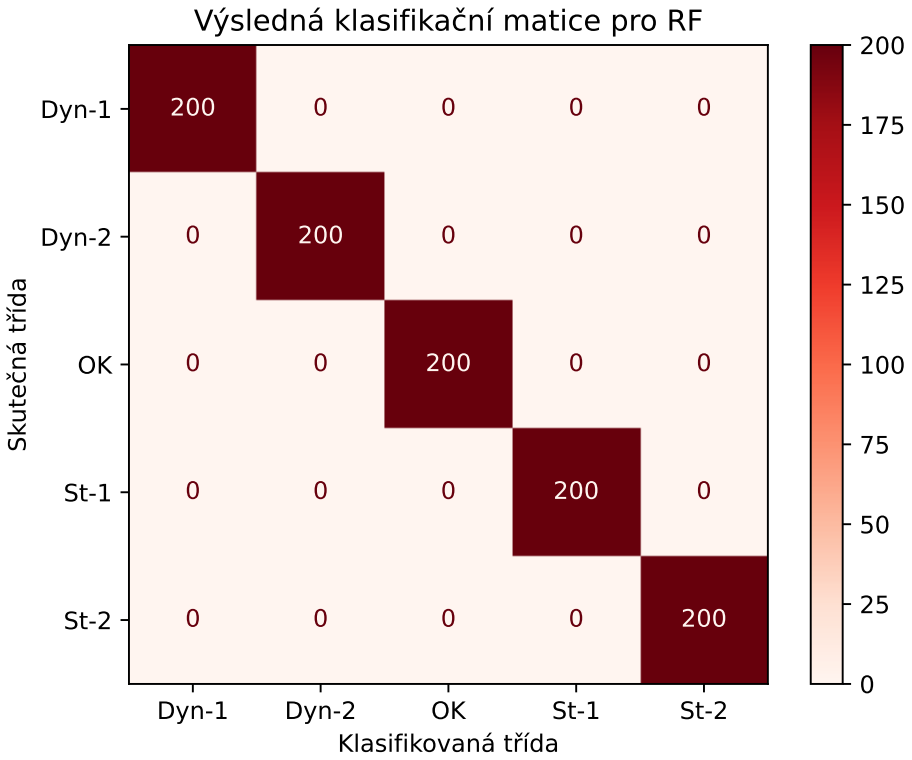
```
RandomForestClassifier(criterion='gini', max_features='auto',
                       n_estimators=10)
```

Celkem bylo desetkrát testováno učení dat, jehož smyslem bylo statisticky určit přesnost při učení, výsledky jsou zaznamenány v tabulce 5.4. Bylo dosaženo 100,00±0,00 % úspěšnosti při učení.

Tab. 5.4: Záznam testování učení modelu RF

učení	**přesnost**
1	1,000
2	1,000
3	1,000
4	1,000
5	1,000
6	1,000
7	1,000
8	1,000
9	1,000
10	1,000

[9]Tento klasifikátor je podrobně popsán na: `https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html`



Obr. 5.10: Výsledná klasifikační matice pro RF

Na obrázku 5.10 je zobrazena výsledná klasifikační matice pro model náhodného lesa. Testovány byly vzorky dat, které model nikdy „neviděl“, tj. rozdílný datový soubor, než na kterém byl model „učen“. Všem klasifikovaným vzorkům model správně přiřadil jejich skutečnou třídu. Tento model má úspěšnost klasifikace 100 %. Umí rozpoznat OK stav od poruchového. V rámci poruchového stavu dále umí úspěšně rozpoznat o jakou poruchu se jedná (statická nebo dynamická i úroveň jedna nebo dvě).

6 HODNOCENÍ VÝSLEDKŮ

Z naměřených dat vibrodiagnostického modelu pro simulaci poruchy rotačního stroje, konkrétně statické a dynamické nevývahy ve dvou úrovních, byly z časové oblasti spočítány prediktory (statistika). Prediktory, které dostatečně (dokázáno dle výsledků) a jednoznačně popisují simulované poruchy jsou efektivní hodnota (RMS), výběrová směrodatná odchylka (STD), analýza hlavních komponent (PCA) a fázový posun (FP). Pomocí těchto prediktorů dokázaly klasifikační modely s určitou úspěšností určit třídu posuzovaných vzorků. Celkový datový soubor vzorků činil 5000 jednotek, ty byly rozděleny v poměru 80:20 – na 4000 vzorcích se modely učily a 1000 vzorků bylo použito pro ověření, respektive klasifikaci.

U každé metody bylo provedeno hledání nejvhodnějších parametrů, které nejlépe vystihovaly řešený problém. K tomu byla využita funkce `GridSearchCV`, nejlepší parametry byly vypsány u každé použité metody. Hledání nejlepších parametrů bylo téměř bez komplikací (vyjma metody SVM). Bylo nezbytné si projít dokumentaci k jednotlivým metodám na oficiálních stránkách knihovny `Scikit-learn`. Metoda SVM kupodivu vykazovala obrovskou chybovost při snaze nalézt nejlepší parametry. V některých fázích výpočtu se celý program zasekl a nepokračoval dále – tento problém nebyl způsoben délkou výpočtu. S vedoucím práce byl tento problém diskutován a byla snaha provést tento výpočet na fakultní výpočetní technice, která je mnohem výkonnější než autorova domácí. Bohužel se stejný problém objevoval i na těchto počítačích. Problém byl částečně vyřešen spouštěním hledání „po částech“, kdy se odděleně prohledávaly jednotlivé jádrové funkce. Výsledky se zaznamenávaly a následně bylo zvoleno nejlepší dosažené nastavení klasifikačního modelu SVM. U zbylých metod nebyla identifikována žádná komplikace. Všechny metody dosahovaly 100% úspěšnosti při učení[1].

Z aplikovaných metod, metoda KNN měla nejnižší celkovou úspěšnost klasifikace a to 99,10 % – dobrý výsledek. Chyba klasifikace byla u předložených vzorků Dyn-1, kdy v jednom případě klasifikovala Dyn-2 a ve druhém případě OK stav (třídu). Dále u předloženého vzorku Dyn-2 došlo v pěti případech k chybné klasifikaci jako Dyn-1. Všechny tyto chyby klasifikace můžeme označit za nepříliš podstatné, poněvadž došlo „pouze“ k chybnému určení stupně dynamické nevývahy. Stejně tak u předloženého vzorku St-2, byla chybná klasifikace St-1. Mírně znepokojující byla chyba při klasifikaci předloženého vzorku OK jako Dyn-2 – zde nám model tvrdí, že bezchybný stav má dynamickou nevývahu druhého stupně. Stejná chyba klasifikace byla také u modelu metody SVM. Lze usoudit, že se může jednat o „velmi“ odlehlý (specifický) vzorek, který může být zkreslen šumem nebo okolními vlivy při simulaci. Úspěšnost klasifikace OK stavu činila 99,5 %. Jak u metody KNN, tak u metody SVM. Na rozdíl od metody KNN, kdy celková úspěšnost byla 99,10 %, metoda SVM měla celkovou úspěšnost 99,90 %. Metody DT a RF dosáhly

[1] Více než úspěšnost učení nás zajímá stabilita učení. V případě, že by nám úspěšnost při učení nevycházela stále 100 %, pravděpodobně by docházelo k jistému kolísání hodnoty úspěšnosti učení modelu – naším záměrem je mít model, který se učí s co nejvyšší stabilitou.

shodné celkové úspěšnosti a to 100 %.

Způsoby jak zlepšit celkovou úspěšnost klasifikace modelů (rozlišovací schopnost). V případě použitých metod v této práci jde pouze o modely KNN a SVM, do modelů DT a RF netřeba zasahovat (seřazeno sestupně podle pravděpodobnosti zlepšení výsledku):

1. Přidat další prediktory – metoda KNN má problém (minimální) v rozlišení, především dynamické nevývahy prvního a druhého stupně. Zaměřit se na tento problém. Tento způsob je nejvhodnější.
2. Zvýšit počet trénovacích vzorků, model by měl širší portfólio dat, z kterých by určoval predikce.
3. Další optimalizace samotného modelu. Hledání nejlepších parametrů pro model bylo provedeno v určitém rozsahu. Je možné, že existují vhodnější parametry.

7 ZÁVĚR

Diplomová práce pojednává o možnostech využití metod umělé inteligence pro vyhodnocení poruch strojního zařízení. Těžiště tvoří popis použitého programovacího jazyka Python a použitých knihoven (`Scikit-learn`, `SciPy`, `Pandas` ...) k aplikaci metod strojového učení na reálných, naměřených datech z vibrodiagnostického modelu. V rámci popisu klasifikovaných dat byl tento model popsán.

V teoretické části je popsána a definována technická diagnostika, diagnostický prostředek, systém a automatizovaný diagnostický řetězec, který byl aplikován v praktické části. V rámci rešerše informací o umělé inteligenci byly definovány pojmy strojové učení a hluboké učení. Strojovému učení bylo věnováno nejvíce prostoru, protože řešení zadaného problému je jeho součástí a konkrétně jde o klasifikace dat metodou učení s učitelem.

Rozsáhlá kapitola je věnována softwarovým prostředkům, potenciálním alternativám (MATLAB, TensorFlow, PyTorch, Keras ...), ve kterých by bylo možné zadaný problém vyřešit. Téměř všechny popisované softwary jsou volně dostupné (bezplatné) a značnou část z nich využívají velké nadnárodní společnosti jakou jsou CocaCola, Intel, Twitter, Siemens a další. Bylo popsáno programovací prostředí Spyder, ve kterém byla práce řešena a jeho programovací jazyk Python. Při vytváření skriptů (aplikací) v programovacím jazyku Python jsou využívány vytvořené knihovny a moduly, které výrazně ulehčují práci programátorovi. Těmto knihovnám je věnována rozsáhlá pozornost, u některých je přiložen příklad psaní kódu. Důležitou knihovnou, která obsahuje balíčky s klasifikačními metodami je knihovna `Scikit-learn`. Pomocí této knihovny byly vytvořeny všechny modely pro klasifikaci dat. Součástí této knihovny je i balíček `plot_confusion_matrix` pomocí kterého byla vytvořena vizualizace klasifikační matice. Je popsán způsob i s příkladem, jak se orientovat v této matici a jaké informace z ní lze vyčíst nebo si je snadno vypočítat.

V praktické části jsou aplikovány metody klasifikace metodou nejbližšího souseda (KNN), metoda podpůrných vektorů (SVM), rozhodovací stromy (DT) a klasifikace metodou náhodného lesa (RF). Metoda KNN dosahuje celkové úspěšnosti 99,10 %, metoda SVM 99,90 %, metody DT a RF shodně 100 %. Podrobné hodnocení výsledků je uvedeno v kapitole 6.

Mezi přínosy této diplomové práce patří shrnutí a ucelené pojednání o umělé inteligenci, jejích součástí a konkrétní příklad aplikace v technické diagnostice. Součástí je podrobná dokumentace a vytvořené skripty (aplikace) metod klasifikace dat a vizualizace výsledků pomocí výsledné klasifikační matice. Tato reálná aplikace může sloužit jako návod postupu vytvoření funkční umělé inteligence, zaměřené na klasifikaci dat s vysokou celkovou úspěšností. Dílčí postupy jsou podrobně komentovány s přiloženými odkazy na webové stránky s podrobným popisem parametrů jednotlivých metod nebo použitých balíčků z knihoven pro programovací jazyk Python.

Literatura

[1] LEGÁT, Václav. *Management a inženýrství údržby*. Druhé doplněné vydání. Praha: Kamil Mařík - Professional Publishing, 2016, 622 s. ISBN 978-80-7431-163-5.

[2] ČSN EN ISO 9000. *Systémy managementu kvality - Základní principy a slovník*. Praha: Úřad pro technickou normalizaci, metrologii a státní zkušebnictví, 2016. Třídící znak 010300.

[3] MAREK, J. Teorie a stavba výrobních systémů [Přednášky]. Brno: VUT FSI, 2020

[4] ZUTH, D. Technická diagnostika I [Přednášky]. Brno: VUT FSI, 2020

[5] KREIDL, Marcel a Radislav ŠMÍD. *Technická diagnostika: senzory, metody, analýza signálu*. Praha: BEN, 2006, 406 s. : il. ISBN 80-7300-158-6.

[6] Dokumentace ke službě Azure Machine Learning: Deep learning vs. machine learning in Azure Machine Learning. *Microsoft* [online]. [cit. 26. 4. 2021]. Dostupné z: `https://docs.microsoft.com/cs-cz/azure/machine-learning/concept-deep-learning-vs-machine-learning`

[7] HAMMER, Miloš. *Metody umělé inteligence v diagnostice elektrických strojů*. Praha: BEN - technická literatura, 2009, 399 s. : il. ISBN 978-80-7300-231-2.

[8] TAULLI, Tom. *Artificial intelligence basics: a non-technical introduction*. New York: Apress, 2019, xii, 187 stran : grafy. ISBN 978-1-4842-5027-3.

[9] SEJNOWSKI, Terrence J. *The deep learning revolution*. Cambridge: The MIT Press, 2018, x, 342 stran : ilustrace. ISBN 978-0-262-03803-4.

[10] What Is Machine Learning?? A Key For The Beginners. *SEOMAG* [online]. 2019 [cit. 26. 4. 2021]. Dostupné z: `https://bigdataevo.blogspot.com/2019/07/what-is-machine-learning-key-for.html`

[11] WEIDMAN, Seth. *Deep learning from scratch: building with Python from first principles*. Beijing: O'Reilly, 2019, xiv, 235 pages : ilustrace ; 24 cm. ISBN 978-1-492-04141-2.

[12] Supervised Learning vs Unsupervised Learning. Which is better? Lawtomated [online]. 2020 [cit. 28. 4. 2021]. Dostupné z: `https://lawtomated.com/supervised-vs-unsupervised-learning-which-is-better/`. Obrázek.

[13] DAŇHELOVÁ, Jana. *Zpětnovazební učení pro řešení herních algoritmů* [online]. Brno, 2018, 48 s. Bakalářská práce. Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací. Vedoucí práce: Ing. Martin Kolařík, [cit. 28. 4. 2021] Dostupné z: `https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=175619`

[14] MEHTA, Anukrati. Semi-Supervised Learning: An Ultimate Guide To Understanding Semi-Supervised Learning. *Digital Vidya* [online]. 2019 [cit. 28. 4. 2021]. Dostupné z: https://www.digitalvidya.com/blog/semi-supervised-learning/. Obrázek.

[15] *TensorFlow* [online]. [cit. 2. 5. 2021]. Dostupné z: https://www.tensorflow.org/

[16] *PyTorch* [online]. [cit. 2. 5. 2021]. Dostupné z: https://pytorch.org/features/

[17] *Keras* [online]. [cit. 2. 5. 2021]. Dostupné z: https://keras.io/

[18] *KNIME* [online]. [cit. 2. 5. 2021]. Dostupné z: https://www.knime.com/software-overview

[19] *GNU Octave* [online]. [cit. 2. 5. 2021]. Dostupné z: https://www.gnu.org/software/octave/

[20] *What Is MATLAB?*, MathWorks [online]. [cit. 2. 5. 2021]. Dostupné z: https://www.mathworks.com/discovery/what-is-matlab.html

[21] *Machine Learning*, MathWorks [online]. [cit. 2. 5. 2021]. Dostupné z: https://www.mathworks.com/solutions/machine-learning.html?s_tid=hp_brand_machine

[22] What is NumPy? *NumPy* [online]. The SciPy community [cit. 1. 5. 2021]. Dostupné z: https://numpy.org/doc/stable/user/whatisnumpy.html

[23] VIKTORIN, Petr a Miro HRONČOK. Analýza dat v Pythonu. *Nauč se Python!* [online]. Praha: ČVUT, 2017 [cit. 1. 5. 2021]. Dostupné z: https://naucse.python.cz/lessons/intro/pandas/

[24] HUNTER, John D. Matplotlib. *History* [online]. 2008 [cit. 1. 5. 2021]. Dostupné z: https://matplotlib.org/stable/users/history.html

[25] KRESTIANKOVÁ, Tamara. *Získávání znalosti z dat v jazyce Python*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Doc. Ing. Jaroslav Zendulka, CSc, [cit. 1. 5. 2021] Dostupné z: https://www.vutbr.cz/www_base/zav_prace_soubor_verejne.php?file_id=181401

[26] BROWNLEE, Jason. Machine Learning Mastery. *A Gentle Introduction to Scikit-Learn: A Python Machine Learning Library* [online]. 2014 [cit. 1. 5. 2021]. Dostupné z: https://machinelearningmastery.com/a-gentle-introduction-to-scikit-learn-a-python-machine-learning-library/

[27] ZUTH, Daniel a Tomáš MARADA. *Using Artificial Intelligence to Determine the Type of Rotary Machine Fault* [online]. MENDEL. 2018, 24(2), 49-54 [cit. 18. 4. 2021]. ISSN 2571-3701. Dostupné také z: https://doi.org/10.13164/mendel.2018.2.049

[28] ZHENG, Alice a Amanda CASARI. *Feature engineering for machine learning: principles and techniques for data scientists*. Sebastopol: O'Reilly Media, 2018, xiii, 200 stran : ilustrace. ISBN 978-1-491-95324-2.

[29] NILSSON, Nils J. *INTRODUCTION TO MACHINE LEARNING* [online]. Stanford: Stanford University, 1998 [cit. 8. 5. 2021]. Dostupné z: http://robotics.stanford.edu/~nilsson/MLBOOK.pdf

[30] ROBINSON, Scott. *K-Nearest Neighbors Algorithm in Python and Scikit-Learn*. Stack Abuse [online]. 2018 [cit. 8. 5. 2021]. Dostupné z: https://stackabuse.com/k-nearest-neighbors-algorithm-in-python-and-scikit-learn/

[31] Scikit-learn. *Support Vector Machines* [online]. [cit. 12. 5. 2021]. Dostupné z: https://scikit-learn.org/stable/modules/svm.html#classification

[32] ROBINSON, Scott. Stack Abuse. *Decision Trees in Python with Scikit-Learn* [online]. 2018 [cit. 12. 5. 2021]. Dostupné z: https://stackabuse.com/decision-trees-in-python-with-scikit-learn/

[33] Scikit-learn. *Plot the decision surface of a decision tree on the iris dataset* [online]. [cit. 12. 5. 2021]. Dostupné z: https://scikit-learn.org/stable/auto_examples/tree/plot_iris_dtc.html. Obrázek.

[34] Scikit-learn. *Decision Trees* [online]. [cit. 12. 5. 2021]. Dostupné z: https://scikit-learn.org/stable/modules/tree.html

[35] KLASCHKA, Jan a Emil KOTRČ. *Klasifikační a regresní lesy*. In: ROBUST 2004: Sborník prací 13. letní školy JČMF ROBUST 2004 uspořádané Jednotou českých matematiků a fyziků za podpory KPMS MFF UK a České statistické společnosti ve dnech 7. – 11. června 2004 v Třešti. Praha: Jednota českých matematiků a fyziků, 2004. ISBN 80-7015-972-3 [online]. [cit. 12. 5. 2021] Dostupné také z: https://www.statspol.cz/robust/robust2004/klaschka.pdf

[36] YIU, Tony. Towards data science. *Understanding Random Forest: How the Algorithm Works and Why it Is So Effective* [online]. 2019 [cit. 13. 5. 2021]. Dostupné z: https://towardsdatascience.com/understanding-random-forest-58381e0602d2

[37] MALIK, Usman. Stack Abuse. *Random Forest Algorithm with Python and Scikit-Learn* [online]. 2018 [cit. 13. 5. 2021]. Dostupné z: https://stackabuse.com/random-forest-algorithm-with-python-and-scikit-learn/

Seznam symbolů a zkratek

ČSN EN ISO Česká verze mezinárodní normy převzaté evropskou komisí pro normalizaci

UI (*AI*) Umělá inteligence (*Artificial Intelligence*)

GPU Grafický procesor (*Graphic Processing Unit*)

API Rozhraní pro programování aplikací (*Application Programming Interface*)

MEMS Akcelerometr (typ) (*Micro Electro Mechanical Systems*)

St-1 Statická nevývaha první úrovně (třída)

St-2 Statická nevývaha druhé úrovně (třída)

Dyn-1 Dynamická nevývaha první úrovně (třída)

Dyn-2 Dynamická nevývaha druhé úrovně (třída)

OK Bezvadný stav (třída)

RMS Efektivní hodnota (*Root Mean Square*)

STD Výběrová směrodatná odchylka (*Standard Deviation*)

PCA Analýza hlavních komponent (*Principial Component Analysis*)

FP Fázový posun (*Cross Correlation*)

KNN Klasifikace metodou nejbližšího souseda (*K-Nearest Neighbors*)

SVM Metoda podpůrných vektorů (*Support Vector Machines*)

DT Rozhodovací stromy (*Decision Trees*)

RF Klasifikace metodou náhodného lesa (*Random Forests Classifiers*)

Seznam obrázků

Seznam tabulek

Seznam příloh

A Obsah elektronické přílohy

Tato příloha obsahuje naměřená vstupní data v adresáři `statistiky` – `DATA_mereni.zip`. Před spuštěním skriptu na výpočet statistiky je nutno „rozbalit“. Dále jsou v jednotlivých adresářích umístěné skripty pro aplikované metody v praktické části. Každý adresář obsahuje shodné `data_overeni` a `data_uceni`, tj. prediktory, které byly vypočteny ve statistice. `*_model.sav` je uložený model umělé inteligence pro danou metodu, který slouží k predikci tříd.

- `Priloha` ..... kořenový adresář přiloženého archivu
  - `statistika` ..... naměřená data, skript pro výpočet statistiky
    - `DATA_mereni.zip`
    - `statistika.py`
    - `slouceni.py`
  - `SVM` ..... Metoda podpůrných vektorů: data, skripty, model
    - `data_overeni.txt`
    - `data_uceni.txt`
    - `SVM_hledani_parametru.py`
    - `SVM_klasifikace.py`
    - `SVM_test_presnosti.py`
    - `SVM_model.sav`
  - `RF` ..... Klasifikace metodou náhodného lesa: data, skripty, model
    - `data_overeni.txt`
    - `data_uceni.txt`
    - `RF_hledani_parametru.py`
    - `RF_klasifikace.py`
    - `RF_test_presnosti.py`
    - `RF_model.sav`
  - `KNN` ..... Klasifikace metodou nejbližší souseda: data, skripty, model
    - `data_overeni.txt`
    - `data_uceni.txt`
    - `KNN_hledani_parametru.py`
    - `KNN_klasifikace.py`
    - `KNN_test_presnosti.py`
    - `KNN_model.sav`
  - `DT` ..... Rozhodovací stromy: data, skripty, model
    - `data_overeni.txt`
    - `data_uceni.txt`
    - `DT_hledani_parametru.py`
    - `DT_klasifikace.py`
    - `DT_test_presnosti.py`
    - `DT_model.sav`